UNIVERSITY OF GENOVA

PHD PROGRAM IN BIOENGINEERING AND ROBOTICS

Spoken command recognition for robotics

by

Bertrand Higy

Thesis submitted for the degree of *Doctor of Philosophy* (31° cycle)

December 2018

Giorgio Metta, *Istituto Italiano di Tecnologia* — Supervisor
Leonardo Badino, *Istituto Italiano di Tecnologia* — Supervisor

Thesis Reviewers:

Daniele Falavigna, *Fondazione Bruno Kessler* — Reviewer
Paweł Świętojański, *University of New South Wales* — Reviewer

Thesis Jury:

Angelo Cangelosi, *University of Manchester* — External examiner
Daniele Falavigna, *Fondazione Bruno Kessler* — External examiner
Markus Vincze, *Technische Universität Wien* — External examiner

To my parents, who always gave me freedom
to make my own decisions and choose who I wanted to be.

To Coralie and Eliott for their encouragements and love,
and for making this stressful period much more enjoyable.

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. This dissertation contains fewer than 65,000 words including appendices, bibliography, footnotes, tables and equations and has fewer than 150 figures.

Bertrand Higy
March 2019

Acknowledgements

I first would like to express my gratitude to Giorgio Metta, for the enthusiasm he expressed when I first contacted him, as I was looking for a PhD position. It is thanks to him that I joined iCub Facility. Despite all his responsibilities, he has always been available when I needed him.

I then would like to thank Leonardo Badino who supervised my daily work. Thank you for all the time you took for me and for supporting me through this adventure. You have been very comprehensive in a period of my life that was pretty hectic. I learned a lot from you and the rest of the group. Many thanks to all the members of the ASR group, for the engaging conversations.

I'm also very grateful to Vadim Thikhanoff for his help with experiments and other technical delights. You have always demonstrated interest and support, even after hundreds of video shots with a less than cooperative robot :).

Many thanks to Peter Bell for receiving me at CSTR. It has been a great pleasure to work under your supervision. My time in Edinburgh has been a very enriching experience. To all CSTR members, thank you for your friendliness and for making the place such a dynamic environment. A special mention to Ondrej, Joachim and Joanna for their help with the Speech Commands dataset and to Sameer for the very fruitful discussions on sequence-to-sequence models. Thanks also to all occupants of room 3.32 for their warm welcome.

My gratitude also goes to my two reviewers, Daniele Falavigna and Paweł Świętojański, for their insightful comments and suggestions.

I would like to thank all members of iCub facility for making my working environment so enjoyable. A special thought for my closest work mates who will recognize themselves. It is never easy to start a new life abroad, but you helped me feel at home in Genova during the last years.

Finally, I would like to thank my family and friends for their unfailing support throughout this period.

Abstract

In this thesis, I investigate spoken command recognition technology for robotics. While high robustness is expected, the distant and noisy conditions in which the system has to operate make the task very challenging. Unlike commercial systems which all rely on a "wake-up" word to initiate the interaction, the pipeline proposed here directly detect and recognizes commands from the continuous audio stream. In order to keep the task manageable despite low-resource conditions, I propose to focus on a limited set of commands, thus trading off flexibility of the system against robustness.

Domain and speaker adaptation strategies based on a multi-task regularization paradigm are first explored. More precisely, two different methods are proposed which rely on a tied loss function which penalizes the distance between the output of several networks. The first method considers each speaker or domain as a task. A canonical task-independent network is jointly trained with task-dependent models, allowing both types of networks to improve by learning from one another. While an improvement of 3.2% on the frame error rate (FER) of the task-independent network is obtained, this only partially carried over to the phone error rate (PER), with 1.5% of improvement. Similarly, a second method explored the parallel training of the canonical network with a privileged model having access to i-vectors. This method proved less effective with only 1.2% of improvement on the FER.

In order to make the developed technology more accessible, I also investigated the use of a sequence-to-sequence (S2S) architecture for command classification. The use of an attention-based encoder-decoder model reduced the classification error by 40% relative to a strong convolutional neural network (CNN)-hidden Markov model (HMM) baseline, showing the relevance of S2S architectures in such context. In order to improve the flexibility of the trained system, I also explored strategies for few-shot learning, which allow to extend the set of commands with minimum requirements in terms of data. Retraining a model on the combination of original and new commands, I managed to achieve 40.5% of accuracy on the new commands with only 10 examples for each of them. This scores goes up to 81.5% of accuracy with a larger set of 100 examples per new command. An alternative strategy, based on model adaptation achieved even better scores, with 68.8% and 88.4% of accuracy with 10

and 100 examples respectively, while being faster to train. This high performance is obtained at the expense of the original categories though, on which the accuracy deteriorated. Those results are very promising as the methods allow to easily extend an existing S2S model with minimal resources.

Finally, a full spoken command recognition system (named iCubrec) has been developed for the iCub platform. The pipeline relies on a voice activity detection (VAD) system to propose a fully hand-free experience. By segmenting only regions that are likely to contain commands, the VAD module also allows to reduce greatly the computational cost of the pipeline. Command candidates are then passed to the deep neural network (DNN)-HMM command recognition system for transcription. The VoCub dataset has been specifically gathered to train a DNN-based acoustic model for our task. Through multi-condition training with the CHiME4 dataset, an accuracy of 94.5% is reached on VoCub test set. A filler model, complemented by a rejection mechanism based on a confidence score, is finally added to the system to reject non-command speech in a live demonstration of the system.

Table of contents

List of publications

- Bertrand Higy, Alessio Mereta, et al. (2018). "Speech Recognition for the iCub Platform". In: *Frontiers in Robotics and AI* 5. ISSN: 2296-9144. DOI: 10.3389/frobt.2018.00010
- Arman Savran et al. (2018). "Energy and Computation Efficient Audio-visual Voice Activity Detection Driven by Event-cameras". In: *Proceedings of 13th IEEE International Conference on Automatic Face Gesture Recognition*. Xi'an, China
- Bertrand Higy and Peter Bell (2018). "Few-shot learning with attention-based sequence-to-sequence models". In: arXiv: 1811.03519

List of figures

List of tables

Acronyms

Adam Adaptive moment estimation.
API Application programming interface.
ASR Automatic speech recognition.

BiLSTM Bilateral long short-term memory.
BPTT Backpropagation through time.

CD Context dependent.
CHiME Computational Hearing in Multisource Environments.
CI Context independent.
CNN Convolutional neural network.
CSTR Centre for Speech Technology Research.
CTC Connectionist temporal classification.
CTNSC Center for Translational Neurophysiology of Speech and Communication.
CUED Cambridge University Engineering Department.

DARPA Defense Advanced Research Projects Agency.
DCT Discrete cosine transform.
DD Domain-dependent.
DI Domain-independent.
DNN Deep neural network.
DOF Degrees of freedom.

E2E End-to-end.
EBNF Extended Backus-Naur form.
EM Expectation maximization.
ESPnet End-to-End Speech Processing Toolkit.

FER Frame error rate.

fMLLR Feature-space maximum likelihood linear regression.
FT Fourier transform.

GAN Generative adverserial network.
GD Gradient descent.
GMM Gaussian mixture model.
GRU Gated recurrent unit.

HMM Hidden Markov model.
HTK Hidden Markov Model Toolkit.

IIT Italian Institute of Technology.
IOL Interactive object learning.
IoT Internet of Things.

KLD Kullback–Leibler divergence.
KWS Keyword spotting.

LDA Linear discriminant analysis.
LHN Linear hidden network.
LHUC Learning hidden units contribution.
LIN Linear input network.
LM Language model.
LON Linear output network.
LSTM Long short-term memory.
LVCSR Large-vocabulary continuous speech recognition.

MBR Minimum Bayes risk.
MFCC Mel-frequency cepstrum coefficient.
MIT Massachusetts Institute of Technology.
MLLT Maximum likelihood linear transform.
MMI Maximum mutual information.
MPE Minimum phone error.
MTL Multi-task learning.

NMT Neural machine translation.

OOV Out-of-vocabulary.

PER Phone error rate.

ReLU Rectified linear unit.
RM Resource Management.
RNN Recurrent neural network.
RNN-T Recurrent neural network transducer.
ROC Receiver operating characteristic.
ROS Robot Operating System.
RPC Remote procedure call.

S2S Sequence-to-sequence.
SAT Speaker adaptive training.
SAwT Speaker-aware training.
SC Speech Commands.
SD Speaker-dependent.
SDR Spoken document retrieval.
SER Sentence error rate.
SGD Stochastic gradient descent.
SI Speaker-independent.
sMBR State-level minimum Bayes risk.
SNR Signal-to-noise ratio.
STL Single-task learning.
SVM Support vector machine.

TI Texas Instruments.

UBM Universal background model.

VAD Voice activity detection.
VoCub Vocal Commands for iCub.
VTLN Vocal tract length normalization.

WER Word error rate.
WSJ Wall Street Journal.

YARP Yet Another Robot Platform.

Chapter 1

Introduction

Over the last years, automatic speech recognition (ASR) technology has progressed dramatically. However, despite some recent claims, it is far from being a solved problem. If it is true that human-level performance has been reached on some datasets (e.g. Switchboard) (Xiong et al., 2016), there are still many challenges for the ASR community to solve.

Hence, ASR technology is very effective in closed-microphone conditions, where the signal-to-noise ratio (SNR) is high. A very good example is the wide availability of audio assistants (e.g. Siri, Google Now, Cortana) on smartphones nowadays. Though, in distant microphone and noisy environment conditions, speech recognition is still a challenging task and an object of active research. Those two elements – distance and noise – are strongly related: the more you increase the distance between the microphone and the speaker, the stronger is the impact of the noise and reverberation on the recorded signal and the harder is the task.

The goal gets even harder with continuous listening conditions, as considered in command recognition. The additional constraint of detecting when the commands are pronounced may add many false positives against which the model should be robust. This is to contrast with most of the work in ASR which takes as assumption that the sentences are already segmented.

Spoken command recognition for robotics, where commands should be segmented from a continuous audio stream in noisy conditions and with a possibly distant microphone is thus very challenging and the main problem addressed in this thesis.

To illustrate the difficulty of the task, a comparison can be made with an application that already reached the market and which has to deal with very similar conditions – smart speakers (e.g. Google Home or Amazon Echo). These home assistants also work with distant microphones and perform continuous listening. However, in practice, the system really attends continuously to a single word only, called "wake-up" word ("Ok Google" and "Alexa"

for the two systems respectively), the detection of which activates the speech recognition system that will recognize the actual command. This strategy allows for much more robust performance as the system has only to discriminate between two classes: keyword present or keyword absent. By gathering a very large number of examples for the wake-up word, the number of false positives can then be reduced to a minimum. Despite that, it can be seen from the news that those devices still recognize commands wrongly sometimes and, e.g., can send a private conversation between two persons to one of their contacts without their awareness[1].

This example highlights the difficulty of the task. My objective was to move one step further and perform continuous command recognition without resorting to a wake-up word. As large vocabulary command recognition would have been too challenging, I decided to focus on small vocabulary tasks. This fits well with robotic applications where the number of actions a robot can perform is still quite limited usually.

A smaller number of commands also makes their classification easier, which compensates for another limitation we have: the shortage of resources. I did not have the means of gathering a big dataset to train robust acoustic models on a large vocabulary and existing corpuses do not match the runtime conditions of my application. Using a smaller vocabulary is thus a first way to counterbalance the negative impact of the scarcity of data. An additional possibility I explore in this thesis is the use of domain adaptation techniques (chapter 4) that allow to leverage on bigger existing dataset to get more robust acoustic models.

However, one of the main limitation of this small vocabulary approach is its lack of flexibility. As we will see, it is hard to extend the vocabulary of the command recognition system beyond its original scope without impacting the performance or requiring the collection of additional data. Working on low resource scenarios is a first way to mitigate this issue, as it reduces the effort required to gather examples of new commands. I additionally explore the possibility of using few-shot learning strategies to extend the vocabulary with a minimum number of examples for each new command (chapter 5). I combine this later work with the use of end-to-end (E2E) models in order to simplify the training and deployment procedures, so as to make this technology accessible to the widest audience possible.

Finally, once a command is recognized, the problem of taking an appropriate action arises. Interpreting the meaning of an utterance is the focus of natural language understanding and is out of the scope of this thesis. A simple one-to-one mapping is used here to assign an action to each command.

To summarize, the work presented here tries to answer following questions:

[1] https://www.theguardian.com/technology/2018/may/24/amazon-alexa-recorded-conversation

- How to build a robust small vocabulary command recognition system in low resource conditions?
- How to compensate for the lack of flexibility of the system and make it easier to extend the vocabulary?
- How can I simplify the use of ASR technology for the robotic community and allow non-expert users to easily train/adapt a model for their own task?

1.1 Characteristics of spoken command recognition for robotics

I will start by defining the main characteristics of the target application and its specificities with respect to other types of scenarios. As I started my work, I identified 3 main requirements of spoken command recognition for robotics: high robustness to noise and distance, online/continuous recognition and computational efficiency. I focused my efforts on the first requirement, robustness, which is key in the user's experience. Though, despite the other two aspects not being the subject of active research, they guided the technical decisions that were made throughout the thesis. Techniques that were too computationally intensive or would lead to high latency were not considered.

A second objective I fixed for my work was to make command recognition technology more accessible to non-expert users. This requirement is not imposed by the task but will greatly augment the impact of the developed technology. Due to their simplicity and intuitiveness, vocal interfaces provide a very appealing way to command all the electronic equipments that surround us. Though, they are not straightforward to build, especially for small groups or companies. Through this thesis, I wanted to help making it easier for anyone to build a command recognition system. Chapter 5, which considers the use of E2E models, is a step in that direction.

1.1.1 Robustness

Robustness is a key property for an application to attain user acceptance level and reach the market. I would argue that command recognition for robotics is even more dependent on this factor. While the consequences of a misrecognition are very limited and usually reversible in the case of, e.g., a vocal assistant on a smartphone, a robot can affect our environment in a more dramatic way. The robustness of the command recognition system is thus critical

if robots have to be used in day-to-day environments (e.g. home, office, hospitals) and alongside humans. This is the reason why robustness is one of the main focuses of my thesis.

The main obstacle to achieving good accuracy in my case is the absence of big datasets matched to the runtime conditions. Few datasets cover distant and noisy situations. Also, the specific kind of noise present in the background is important as a mismatch between training and testing environments may lead to poor performance. Additionally, while datasets usually contain recordings of native-speakers, the users of iCub robot (which is used as a test case) are mainly non-native English speakers.

1.1.2 Continuous listening

A simple strategy to avoid the problem of continuous listening on smartphones could be to use a manual trigger to decide when speech starts and stops. Apart from not being convenient, this strategy is hard to apply for robotic applications. Robots usually do not offer an easy way to signal the sentence end points (such as a button or a tactile interface) and control of the robot is expected to be possible at distance. The same is true of smart devices, where it is now common to use "wake-up" words, i.e. specific keywords that precede and signal the beginning of a command. Wake-up words improve the users experience as they allow hand-free interactions with the device. I tried to go even further by trying to recognize the commands reliably without the use of such additional cue.

To handle the continuous aspect of the task, I decided to resort to a voice activity detection (VAD) system, that is responsible for detecting when the input is likely to contain speech (any speech) and activating the ASR system. The use of a VAD module has two main advantages: (1) it filters most of the silence allowing the command recognition system to focus on segments of speech (thus working in conditions closer to the ones usually considered with ASR systems) and (2) it avoids running an expensive command recognition system continuously.

Indeed, another consequence of performing continuous listening is that we need the system to run all the time and online (in the sense of real-time). This adds to the pre-existing computational constraints as we will now see.

1.1.3 Computational requirements

As any embedded ASR system, a command recognizer for a robot has strong computational constraints, in terms of both memory and computation capability. This is even more the case in robotics (compared to, e.g., a smart speaker) as command recognition is not the main task

of the robot and many other functionalities (such as vision, motor control or planning to mention a few) compete for the resources. Obviously, the continuous and online aspects of the task add to the difficulty.

This deployment constraints are sometime in opposition with research objectives, where high performance can be achieved at the expense of such practical concerns. A good example is the use of ensembles, which improves accuracy at a high computational cost. Hence, even though I did not perform active research on less intensive speech recognition technology, techniques requiring heavy computational resource were discarded from the outset.

1.1.4 (Re)Usability

A system which would be polyvalent and robust at the same time is hard to obtain with limited resources (in terms of supervised training data) and with all the constraints I already mentioned. I thus propose to trade the polyvalence for robustness by focusing on application-specific models, where the system is trained for a specific task and hence specific vocabulary and runtime conditions. If we want to increase the reusability of such technology, the key point is then to make it easier for non-expert users to gather data and train a model for their own application.

A first answer to this concern lies in the low resource setup we adopt, which facilitates the collection of data on a per-task basis. Additionally, I explored the possibility of using few-shot learning to expand the vocabulary with low data requirements (see chapter 5). In terms of training and deployment, I explored the adequacy of E2E architectures for our setup as they greatly simplify these procedures.

1.2 Thesis structure and contributions

I will start with an introduction to speech technology in chapter 2, comparing ASR and keyword spotting (KWS) tasks. While they each have their own specificities, the application domain considered here require both functionalities. I will then spend some time to introduce the resources (datasets and toolkits) used throughout the thesis (chapter 3). This includes the Vocal Commands for iCub (VoCub) dataset which has been gathered for this thesis. After this, I will present the work I did on domain adaptation (chapter 4), which addresses the problem of robustness in low resource settings and is the first contribution of this thesis. Chapter 5 will be dedicated to the second main contribution of the thesis, the study of few-shot learning for E2E models. This work tries to address the second and third questions

listed at the beginning of this chapter, related to flexibility and reusability. Chapter 6 presents the development of iCubrec, a fully operational command recognition system for the iCub platform (Metta, Natale, et al., 2010). Together with the VoCub dataset, this system composes the more practical contribution of my work. I will conclude in chapter 7 and discuss possible directions for future work.

Chapter 2

Automatic speech processing

Despite them sharing many characteristics, speech recognition and keyword spotting have both their specificities, explaining why different metrics and solutions are used to solve each task. I will argue that command recognition stands somewhere between them and can thus borrow ideas from both fields. After discussing the differences and the similitudes between these three problems, I will give an overview of the methodologies employed for each of them. I will finish by presenting the overall framework I intend to use on the robot.

2.1 Speech recognition versus keyword spotting

First, I would like to precise what is intended here by speech recognition, as the expression can have a broader understanding. By ASR, I imply the task of speech to text transcription from pre-segmented audio, where the problem is to find the most likely sequence of words $\boldsymbol{w}^* = (w_1^*, \ldots, w_N^*)$ given a sequence of acoustic observations $\boldsymbol{O} = (\boldsymbol{o}_1, \ldots, \boldsymbol{o}_T)$. Traditionally, most of the acoustic corresponds to speech and silence is only present in small proportions. Also, most of the vocabulary present in the sentences is known and out-of-vocabulary (OOV) words are an exception.

Conversely, KWS is concerned with detection of a small set of words in a live stream of audio. Silence then constitutes most of the observed signal, and most of the remaining speech correspond to OOV words. Trying to maximize recognition of the keywords can then easily result in many false detections. The challenge of KWS is to achieve the highest detection rate possible while maintaining the false detections to a minimum. This requires good models of silence and OOV speech.

The standard metrics used for each task reflect this specificities. The word error rate (WER) used in speech recognition (see subsection 2.8.1 for the precise formulation) evaluates

the correctness of the predicted text with respect to a reference, taking substitutions, insertions and deletions into account. For KWS, it is more common to use notions such as false positive and true positive (definitions will be given in subsection 2.8.2), which reflects the constraint of balancing the maximization of keywords detection against false detections in conditions where most of the input does not contain the expected words.

2.2 Spoken command recognition

Similarly to KWS, command recognition is concerned with the detection of commands in a continuous stream of audio. It then faces the same challenges: not only should the commands be distinguished from one another but we additionally need to detect when they happen in a stream containing mainly silence and non-command speech. However, a big difference with respect to KWS is that we do not restrict ourselves to single keywords. Commands can be fully-fledged sentences following a complex grammar. In that respect, the recognition of a command is closer to a speech recognition task with a small vocabulary and a strict grammar.

As a consequence, some strategies used for KWS are not easily transposable to our setup. For example, in one of the work closest to ours, Chen et al. (2014) trained a deep neural network (DNN) to directly predict the probability of the keywords being present in the audio at each time step. They propose a simple strategy to extend this to the detection of short expressions composed of several words. This strategy consists in combining into a final confidence score the probability of each individual word composing the expression (over a fixed window), by simply multiplying them. This posterior handling mechanism may not be adequate for more complex expressions or sentences though. One limit of this mechanism, as acknowledged by the authors, is that it does not enforce the order of the labels in the sequence. It is thus impossible to distinguish sentences like "*Put the glass in the water*" or "*Put the water in the glass*". Also, negative sentences (e.g. "*Don't move the box*") may trigger the recognition of the positive version of the sentence ("*Move the box*"). This is partially solved by Prabhavalkar, Alvarez, et al. (2015) who proposed an improved version of the posterior handling mechanism taking the relative order of words into account.

These examples illustrate the main difference between complex sentences and keywords. Unfortunately, there is no way to handle complex sentences in a simple way that would allow to easily adapt work on KWS. One will inevitably have to resort to more complex representations such as language models, which are used for long in ASR systems.

Thus, command recognition can either be seen as a KWS task with key-phrases in place of keywords, or as an ASR system with strict vocabulary and grammar, operating in live conditions where most of the audio consists of silence and non-command speech.

The following sections will cover a high-level presentation of the methodology used in these tasks.

2.3 Feature extraction

For both tasks, it is quite uncommon to work directly on the acoustic signal. Instead, manually engineered features are usually computed and used in place of the raw signal. These features have properties that make them more suitable for learning a statistical model and were key in achieving good performance, at least until recently. Indeed, the newly introduced DNN models have been shown to be better at extracting useful features from their input. It as been found that neural networks perform better when using mel-scaled log filter bank coefficients rather than the more complex mel-frequency cepstrum coefficients (MFCCs) commonly used with Gaussian mixture models (GMMs) (Mohamed et al., 2012; Li, Yu, et al., 2012). It has further been showed that a neural network can even use the power spectrum (Sainath, Kingsbury, et al., 2013) or the raw signal directly (Palaz et al., 2013).

The two most commonly used feature types, regardless of the kind of task, are filter bank coefficients and MFCCs. To motivate their use, I will start by presenting a few properties of the human perception that inspired them.

2.3.1 Human speech perception

The human ear is responsible for transforming the variations of the air pressure into pulses that can be transmitted to the auditory nervous system for processing. It is sensitive to frequencies between 20 Hz and 20 kHz roughly. Transforming the acoustic pressure signal in the time-domain into a representation in the frequency domain is the role of the cochlea. While performing this conversion, the cochlea organizes the information in filters. Each filter responds maximally to a small range of frequencies and those filters are organized topologically, the filters closest to the cochlear base responding to the highest frequencies while the filters closest to its apex respond to the lowest frequencies. This conversion of the variation of the air pressure over time into the frequency domain and the organization of the representation into band-pass filters is the first property of human's auditory perception.

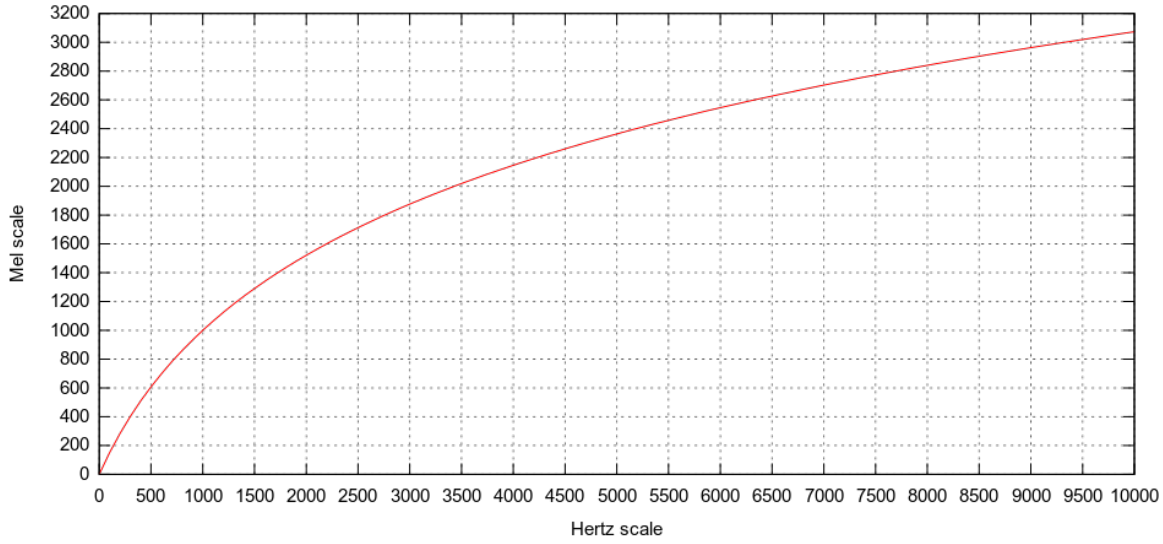

FIGURE 2.1 Perception of pitch (mel scale) versus frequency (hertz scale).
(Krishna Vedala, Mel-Hz plot, CC BY-SA 3.0)

Other two properties that are pertinent to our discussion are the non-linear perception of pitch and loudness. It has been observed that the sensitivity of the human auditory system is not the same across all frequencies. Hence, to elicit the same perception of pitch increase, larger and larger intervals are necessary between the sounds as we go up the frequency scale. The mel scale (represented in Figure 2.1) accounts for this non-linear perception of pitches and is defined as:

$$m = 2595 \log_{10}\left(1 + \frac{f}{700}\right), \tag{2.1}$$

where f is the frequency and m is its equivalent on the mel scale.

Similarly, the perception of loudness by humans is non-linear, and the perceived change in sound intensity gets smaller as the intensity increases. This is exemplified by the dB scale of sound intensity which is also logarithmic.

A deeper introduction to human speech perception can be found in Huang, Acero, et al. (2001, section 2.1.3).

2.3.2 Filter bank coefficients

The mathematical equivalent of the conversion from air pressure variations to frequency by the ear is the Fourier transform (FT). It is usually the first transformation applied to the sound to obtain features. To compute the FT, a small window of signal is required (25 ms traditionally). This process is repeated at fixed step intervals (10 ms usually), the power

spectrum $\boldsymbol{p}_i$ of the i^{th} frame being computed as:

$$\boldsymbol{p}_i = \frac{|FT(\boldsymbol{o}_i)|^2}{N}, \tag{2.2}$$

where $FT(\cdot)$ is the N-point Fourier transform.

Triangular band-pass filters are then applied to the output to extract frequency bands. The filters are equally spaced on the mel scale to account for the non-linear perception of pitch by humans.

Finally, the log function is usually applied to also mimic the non-linear perception of loudness, resulting in log mel-scale filter bank coefficients.

2.3.3 Mel-frequency cepstral coefficients

The filter bank coefficients we just described are highly correlated, which may be an issue. This is especially the case with the GMM-hidden Markov model (HMM) approach where diagonal matrices are often used to model covariance, so as to reduce the number of parameters (growing as the square of the number of coefficient for a full covariance matrix). A discrete cosine transform (DCT) can be additionally applied to the coefficients in order to decorrelate them, leading to the MFCCs (Davis and Mermelstein, 1980).

2.3.4 Common practices

Another common practice is to compute the first and second derivatives of the features and add them to the feature vector. Additionally, for DNNs, a context of several frames is usually used, where the frame's features are appended with features from a few frames before and after it.

2.4 Speech recognition

Speech recognition is a sequence-to-sequence mapping problem where one tries to map from the sequence of acoustic observations $\boldsymbol{O} = (\boldsymbol{o}_1, \ldots, \boldsymbol{o}_T)$ to a sequence of words $\boldsymbol{w} = (w_1, \ldots, w_N)$ (with $N \ll T$). The goal then is to find the most likely sequence of words $\boldsymbol{w}^*$ given the acoustic, out of all possible hypothesis H, that is:

$$\boldsymbol{w}^* = \arg\max_{\boldsymbol{w} \in H} P(\boldsymbol{w}|\boldsymbol{O}) \tag{2.3}$$

It is quite uncommon though to work on the acoustic observations directly. Instead, they are usually preprocessed to obtain features such as the filter bank coefficients or the MFCCs. The set of features computed from the raw sound are then replacing the acoustic observations in Equation 2.3[1].

This problem is usually decomposed using Bayes rule as:

$$\boldsymbol{w}^* = \arg\max_{\boldsymbol{w}\in H} \frac{P(\boldsymbol{O}|\boldsymbol{w})P(\boldsymbol{w})}{P(\boldsymbol{O})} \tag{2.4}$$

$$\propto \arg\max_{\boldsymbol{w}\in H} P(\boldsymbol{O}|\boldsymbol{w})P(\boldsymbol{w}) \tag{2.5}$$

The probability of the features $P(\boldsymbol{O})$ is independent of $\boldsymbol{w}$ during decoding and can thus be ignored. We have two remaining components: (i) $P(\boldsymbol{O}|\boldsymbol{w})$ which estimates the probability of the acoustic based on the sequence of words and is usually referred to as the acoustic model, and (ii) the prior probability $P(\boldsymbol{w})$ which is also known as the language model. The two components are typically estimated independently.

2.4.1 The acoustic model

I will focus here on HMM-based acoustic modeling as it is the dominant approach in ASR.

The GMM-HMM acoustic model

A hidden Markov model is a statistical model where the systems is assumed to follow a Markov process with unobserved states, also called hidden states (illustrated in Figure 2.2). The only access we have to the hidden states is through observed variables which are related to the hidden states via probability distributions.

The HMM is defined by:

- a set $\mathcal{O} = \{\mathcal{O}_1, \ldots, \mathcal{O}_M\}$ of observed variables (discrete or continuous).
- a set $\Omega = \{1, 2, \ldots, K\}$ of hidden states.
- the transition probability matrix $\boldsymbol{A} = \{a_{ij}\}$, where a_{ij} is the probability of following the transition from state i to state j.
- the output probability distribution $B = \{b_i(\boldsymbol{o})\}$, where $b_i(\boldsymbol{o})$ defines the probability of observing output $\boldsymbol{o}$ in state i. Classically in ASR, multivariate GMMs were used to

[1] The notation is overloaded: $\boldsymbol{O}$ refers to both the acoustic observations and the features derived from them.

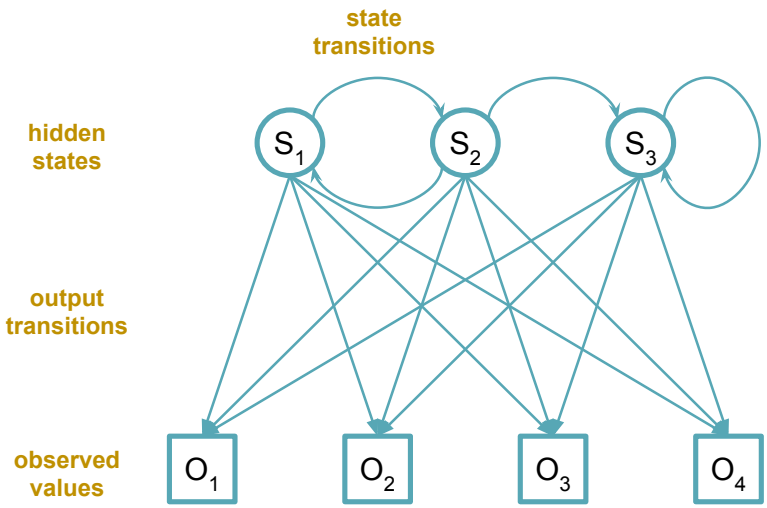

FIGURE 2.2 Illustration of a hidden Markov model with 3 hidden states and 4 discrete observed values.

model the output distribution of continuous variables (see e.g. Rabiner, 1989, for an introduction to the use of GMM-HMM in speech recognition).

- the initial state probability distribution π, which defines the probability of starting in any of the states.

The model parameters $\boldsymbol{\theta} = \{\boldsymbol{A}, B, \pi\}$ can be estimated through the Baum-Welch algorithm (Baum and Petrie, 1966), which is an instance of the expectation maximization (EM) algorithm (Dempster et al., 1977).

As can be seen from this definition, several simplifications are used to make the problem more manageable. First, the HMM is supposed to follow a first order Markov process, that is the transition to a state j only depend on the current state i. Said otherwise, the probability of being in a specific state at step t given the states history, $P(s_t|s_{t-1},\dots,s_1)$, is reduced to the probability of this state given the previous state only, $P(s_t|s_{t-1})$. Also, observations are assumed to be independent of previous observations and states given the current state. Thus, $P(\boldsymbol{o}_t|\boldsymbol{o}_{t-1},\dots,\boldsymbol{o}_1,s_t,\dots,s_1)$ is reduced to $P(\boldsymbol{o}_t|s_t)$. This simplifications, as inaccurate as they are, proved good enough for the problem at hand. The probability of the observed features

can then be formulated as:

$$P(\boldsymbol{O}|w) = \sum_{s \in S} P(\boldsymbol{o}_1|s_1)P(s_1) \prod_{t=2}^{T} P(\boldsymbol{o}_t|s_t)P(s_t|s_{t-1}) \tag{2.6}$$

$$= \sum_{s \in S} b_{s_1}(\boldsymbol{o}_1)\pi(s_1) \prod_{t=2}^{T} b_{s_t}(\boldsymbol{o}_t)a_{s_{t-1},s_t}, \tag{2.7}$$

where S is the set of all state sequences matching the sequence of words w.

From states to words

The only point missing in the above description of the GMM-HMM approach is the link between the hidden states and the sequence of words. The smallest unit in terms of pronunciation is the phoneme, from which we can define the pronunciation of any word. Hence, it would be reasonable to relate states and phonemes in a one-to-one fashion. Though, the pronunciation of a phoneme is not constant over time. A common practice is thus to split each phoneme into several consecutive states (3 states is a popular choice) that allow to model differently the start, middle and end of the phoneme. When the phones are considered independently from one another, we talk about context independent (CI) phonemes or monophones.

It is further known that the pronunciation of a phoneme is influenced by preceding and following ones, a phenomenon known as the co-articulation effect. Context dependent (CD) phonemes (Schwartz et al., 1985), or triphones, which distinguish the pronunciations of a phoneme in different contexts, have been shown to improve acoustic modeling.

Though, CD modeling assumes that all contexts are different, leading to a huge number of triphones. If we consider that English uses about 44 phonemes, we obtain 44^3 triphones, which is hardly manageable. Considering all triphones is very expensive and requires enough data to model each of them accurately. However, some contexts can have a very similar effect on the phoneme. An alternative then is for similar triphones to share statistical parameters (Young and Woodland, 1994; Young, Odell, et al., 1994), where phonetic properties are used to evaluate the similarity. These clustered CD phonemes, also called senones, offer a good trade-off between the exactitude of triphones and the difficulty of learning them reliably.

Using a pronunciation dictionary, which maps words to their phoneme transcription, we can then determine the set S of state sequences corresponding to a sequence of words.

The DNN-HMM acoustic model

The classical GMM-based approach to estimating the output probabilities of the HMM was recently replaced by DNN-based models (Hinton, Deng, et al., 2012). While the use of shallow neural networks was already proposed by Bourlard and Wellekens (1990), the use of deep networks (thanks to recent progress in model optimization) and estimation of their parameters on bigger datasets allowed to surpass the traditional GMM-HMM paradigm.

In order to use DNNs to estimate the output probabilities of the HMM, one has to slightly adapt Equation 2.6 though. Indeed, the DNN estimates $P(s_t|\boldsymbol{o}_t)$ instead of $P(\boldsymbol{o}_t|s_t)$. The two probabilities can be related by following formula:

$$b_{s_t}(\boldsymbol{o}_t) = P(\boldsymbol{o}_t|s_t) = \frac{P(s_t|\boldsymbol{o}_t)P(\boldsymbol{o}_t)}{P(s_t)} \tag{2.8}$$

$$\propto \frac{P(s_t|\boldsymbol{o}_t)}{P(s_t)} \tag{2.9}$$

Again, the probability $P(\boldsymbol{o}_t)$ being independent of s_t (over which we optimize), it can be ignored.

In order to train the neural network to predict $P(s_t|\boldsymbol{o}_t)$, which is done in a supervised manner, one need a reference state s_t for each input $\boldsymbol{o}_t$. Despite some attempts to train DNNs from scratch (Senior, Heigold, et al., 2014; Zhang and Woodland, 2014), the most common strategy is still to start by training a GMM-HMM acoustic model first, which can be used to compute the alignment between the states and the observed frames (and also provide the triphone clustering when senones are used). Strategies to train GMMs from scratch, such as flat start initialization (where the states are evenly spaced across the sound segment) are readily available, and the model can then be refined in an iterative procedure.

To be precise, the method I just presented is referred to as the hybrid DNN-HMM approach. An alternative use of DNNs, called the tandem approach (Hermansky et al., 2000), consists in using the neural network to compute new features that will be used as input to a more traditional GMM-HMM acoustic model. This later approach is not considered in this thesis and whenever I refer to the DNN-HMM approach, the hybrid one is intended.

More details on DNN-based acoustic models will be given in section 2.5.

2.4.2 The language model

The acoustic model estimates the probability of $\boldsymbol{w}$ taking only the acoustic information into account. This can lead to the recognition of sentences that are not valid for the language

considered. The additional constraints imposed by the language are taken into account in the optimization task through the prior $P(\boldsymbol{w})$, also called the language model.

The main problem for many tasks is to model the open-ended set of sentences that should be accepted by the language model. Here again, some approximations are usually used to make the task manageable. The most common approach is to use n-gram models (Damerau, 1971), where the probability of a word is assumed to depend only on a finite number $(n-1)$ of preceding words. Hence, the probability of a sequence of words can be formulated as:

$$P(\boldsymbol{w}) = \prod_{t=1}^{T} P(w_t | w_{t-1}, \ldots, w_{t-n+1}) \tag{2.10}$$

Some recent work proposes to use recurrent neural networks (RNNs) to perform language modeling (Mikolov et al., 2010). The main limitation with the use of neural networks as language model (LM) though, also true for n-grams with a large context, is the computational cost. A good compromise then consist in using a simpler LM model initially (e.g. a 3-gram LM) and rescore the best predictions with a more complex one (e.g. a RNN-LM or a 5-gram LM).

In some cases where a restricted grammar is imposed, such as in the setup considered here, the grammar can be defined in closed form using a formalism such as the extended Backus-Naur form (EBNF) (used to define context-free grammars).

2.4.3 Decoding

Given acoustic and language models, as well as a pronunciation dictionary relating each word to its phonemic transcription, the decoder finally computes the sequence of words with highest probability. The standard decoding algorithm for HMM-based systems is the Viterbi algorithm (Viterbi, 1967). It relies on principles from dynamic programming to efficiently computes the required scores. Instead of a 1-best word sequence, it can also output a lattice of N-best paths.

2.5 Deep neural networks for acoustic modeling

The use of DNNs to estimate state probabilities in the hybrid framework plays an important role in this thesis. It thus deserves to be treated in more details. I will try here to give a general overview of the main architectures, optimization and training strategies used with these models.

2.5.1 Architecture

The most common architecture is the fully connected feed-forward neural network, where each layer of units is fully connected to the next layer and no connection exist between units of a layer or from a layer going backward.

Alternatively, the use of convolutional layers has been explored. Originally proposed for vision (Lecun et al., 1990), it has since been shown that convolutional layers can also be applied successfully to speech recognition, either along the frequency axis alone (Abdel-Hamid et al., 2013), or along both frequency and time (Tóth, 2014). Convolutional neural networks (CNNs) use a small weight matrix spanning only a small patch of input features. To cover the whole input, this patch is repeatedly applied on a shifted window of values. The stride (or shift) used to move the window allows to downsample the input. Additionally, a pooling layer can also be used to reduce the dimensionality. The max-pool operation (based on the maximum function) is a popular choice. Unlike standard DNNs which ignore the input topology, CNNs leverage on the correlations in time and frequency present in the speech. The weight being shared along one or both dimensions, they also offer more compact models and naturally offer translational invariance.

Speech being a sequential signal, recurrent approaches have naturally been considered as well (see e.g Robinson, 1994; Deng et al., 1994). While backpropagation can be applied to their training using a technique called backpropagation through time (BPTT) (Werbos, 1988), one limit of this approach is related to the vanishing/exploding gradient problem (Bengio et al., 1994): as the number of layers through which we backpropagate the error increases, the gradient tends to shrink or grow out of control. This historically limited the depth of DNNs and the applicability of BPTT to long sequences. While some solutions have been developed to avoid this problem, such as careful initialization or rectified linear unit (ReLU) activation function (Nair and Hinton, 2010), special types of recurrent units have also been proposed. The long short-term memory (LSTM) unit (Hochreiter and Schmidhuber, 1997) was the first of this kind and uses a gate mechanism, which allows the gradient to flow more freely through the layers. These types of units have since been shown to be very effective for different sequential tasks including ASR (see e.g. Graves, Jaitly, and Mohamed, 2013; Sak et al., 2014).

Residual layers (He, Zhang, et al., 2016), which were also proposed for vision originally, allowed to reach another level in the number of layers that can effectively be used. The idea stemmed from the observation that using very deep networks can still lead to a degradation of performance compared to shallower ones. He, Zhang, et al. proposed to reformulate layers so as to learn a residual function, which is sumed with the output of the preceding layer, instead

of a direct transformation of this same output. This has since been applied successfully to speech recognition (Xiong et al., 2016).

2.5.2 Training criteria

Framewise training

DNNs are used to model the posterior probability of the HMM states given the acoustic features. Hence, a natural training criterion to train them is the framewise cross entropy loss, defined as:

$$L_{CE} = -\sum_t \ln P(\hat{s}_t|\boldsymbol{o}_t), \tag{2.11}$$

where $P(\hat{s}_t|\boldsymbol{o}_t)$ is the probability predicted by the network for the reference state $\hat{s}_t$ given observation $\boldsymbol{o}_t$.

Sequence-discriminative training

Framewise training completely ignores the sequential nature of speech signal. Sequence-discriminative training criteria have been proposed in order to remedy that. Common sequence-discriminative criteria include maximum mutual information (MMI) (Bahl et al., 1986), minimum phone error (MPE) (Povey, 2004) and state-level minimum Bayes risk (sMBR) (Kaiser et al., 2000; Gibson and Hain, 2006; Povey and Kingsbury, 2007). They all have in common to try optimizing the score of the whole reference sentence over any other possible sequence. To do that, they all leverage on statistics that can be collected from the lattice, the difference being the granularity they consider. MMI works at the whole sentence level by optimizing following loss:

$$L_{MMI} = \log \frac{P(\boldsymbol{O}|\hat{s})^{\kappa} P(\hat{\boldsymbol{w}})}{\sum_w P(\boldsymbol{O}|s)^{\kappa} P(\boldsymbol{w})}, \tag{2.12}$$

where $\hat{\boldsymbol{w}}$ is the reference word sequence and κ is the acoustic scaling factor. At the denominator, we find the sum over all word sequences in the lattice as an approximation of all possible word sequences.

MPE and sMBR are both minimum Bayes risk (MBR) objectives which consider different levels of granularity (the phone and state level respectively). The corresponding criteria is:

$$L_{MBR} = \frac{\sum_w P(\boldsymbol{O}|s)^{\kappa} P(\boldsymbol{w}) A(\boldsymbol{w}, \hat{\boldsymbol{w}})}{\sum_{w'} P(\boldsymbol{O}|s')^{\kappa} P(\boldsymbol{w}')}, \tag{2.13}$$

where $A(\boldsymbol{w}, \hat{\boldsymbol{w}})$ is the raw accuracy of word sequence $\boldsymbol{w}$ with respect to reference sequence $\hat{\boldsymbol{w}}$, that is the number of correct phone or state labels respectively. A comparison of the different criteria has been done by Veselỳ et al. (2013), which showed little difference between them.

More recently, a lattice-free version of the MMI criterion has been proposed (Povey, Peddinti, et al., 2016) which showed improved results.

The connectionist temporal classification loss

Another way to train networks in a sequence-discriminative manner and without the need for pre-computed alignments is the connectionist temporal classification (CTC) loss (Graves, Fernández, et al., 2006). Usually applied on top of a RNN, CTC solves the problem by summing over the probability of all possible alignments corresponding to the expected output sequence.

More formally, consider the input sequence $\boldsymbol{O} = (\boldsymbol{o}_1, \ldots, \boldsymbol{o}_T)$ and an output sequence $\boldsymbol{l} = (l_1, \ldots, l_K)$, where $l_k \in 1, \ldots, L$ with L the number of output labels. An additional blank label – is also considered corresponding to the absence of label.

The probability of sequence $\boldsymbol{l}$ is then defined as the sum over all possible alignments $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_t)$, where $\boldsymbol{\pi}$ allows repetitions of labels and occurrences of the blank label between two different labels of $\boldsymbol{l}$. This is formalized as:

$$P(\boldsymbol{l}|\boldsymbol{O}) = \sum_{\boldsymbol{\pi} \in \phi(\boldsymbol{l})} P(\boldsymbol{\pi}|\boldsymbol{O}), \tag{2.14}$$

where $\phi(\boldsymbol{l})$ is the set of all possible alignments corresponding to $\boldsymbol{l}$. For example, $''aa - b -''$ and $''- a - - b''$ are two valid alignments for $\boldsymbol{l} = ab$ and $T = 5$.

Assuming conditional independence between two consecutive outputs of the network, the probability of alignment $\boldsymbol{\pi}$ can be factorized as follows:

$$P(\boldsymbol{\pi}|\boldsymbol{O}) \approx \prod_{t=1}^{T} P(\pi_t|\boldsymbol{O}), \tag{2.15}$$

where $P(\pi_t|\boldsymbol{O})$ is estimated by the network. By modifying the forward-backward algorithm, the probability $P(\boldsymbol{l}|\boldsymbol{O})$ can be efficiently computed.

The CTC loss can finally be defined as the negative log likelihood of the reference (or ground truth) sequence $\hat{\boldsymbol{l}}$,

$$L_{CTC} = -\ln P(\hat{\boldsymbol{l}}|\boldsymbol{O}). \tag{2.16}$$

Regularization

In addition to the main loss, additional regularization terms can be added to the optimized loss. Regularization terms aim at reducing the overfitting of the model and obtain better generalization. The most common regularization terms are the l_1 and l_2 norms which penalize the norm of the weights and are defined as:

$$L_{l_1} = \| \boldsymbol{\theta} \|_1 = \sum_i |\theta_i|, \tag{2.17}$$

$$L_{l_2} = \| \boldsymbol{\theta} \|_2^2 = \sum_i \theta_i^2, \tag{2.18}$$

where $\boldsymbol{\theta}$ is the set $\{\theta_i\}$ of network's weights.

2.5.3 Optimization procedure

Gradient descent (GD) is without doubt the most popular optimization algorithm, but it is rarely used in its vanilla version. As GD tries to minimize the objective function by computing the gradient over the entire training set, it tends to be very slow. Stochastic gradient descent (SGD) (Bottou, 1998) solves this issue by computing the gradient for each training example and updating the parameters before moving to next example. While this strategy is usually much faster, the updates have high variability which makes the objective function fluctuate. A good compromise then is the mini-batch SGD which takes an intermediate stance: gradient is computed over a small batch of examples, which reduce the variance of the parameter updates while allowing faster training.

These strategies have several limitations though. First, one has to choose a learning rate. This is not a trivial matter as learning rates that are too small can lead to very slow convergence while, conversely, learning rates that are too large will make the loss oscillate or diverge. Learning rate schedules, pre-defined or based on improvement criteria, are often used to improve the learning. However, once again, they have to be defined manually and do not adapt to the evolution of the optimization procedure. Also, the same learning rate is applied to all parameters which is suboptimal.

Several alternative optimization algorithms have been proposed in order to overcome those limitations. A first strategy, used in this thesis, is to add momentum (Rumelhart et al., 1986). Momentum tries to overcome the difficulty of SGD or mini-batch SGD to deal with ravines, that is areas of the objective surface where the curve is much steeper in some directions than in others. To do that, it combines the current gradient with previous updates,

which allows to increase the gradient in direction with constant progress and reduce it in directions where the gradient often changes sign. This can be formalized as:

$$\boldsymbol{v}_t = \gamma \boldsymbol{v}_{t-1} + \eta \nabla_{\boldsymbol{\theta}_{t-1}} J(\boldsymbol{\theta}_{t-1}), \tag{2.19}$$

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} + \boldsymbol{v}_t, \tag{2.20}$$

where $J(\boldsymbol{\theta})$ is the objective function parameterized by $\boldsymbol{\theta}$, $\boldsymbol{v}_t$ is the parameter update at time t, γ is the weight for the momentum term and η is the learning rate.

Adagrad (Duchi et al., 2011) tries to solve the problem of the same learning rate being applied to all parameters. Essentially, the idea is to use higher learning rates for parameters that are infrequently updated and smaller ones for parameter frequently updated. To do so, the algorithm keeps track of the sum of the square of the gradients with respect to each parameter θ_i since the beginning of the training. This is done through the diagonal matrix $\boldsymbol{G}_t \in \mathbb{R}^{d \times d}$, where d is the dimension of the parameter space. The update of parameter θ_i is then defined as:

$$g_{t,i} = \nabla_{\theta_{t,i}} J(\theta_{t,i}) \tag{2.21}$$

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{G_{t,ii} + \varepsilon}} g_{t,i}, \tag{2.22}$$

where $G_{t,ii}$ is the element at position (i,i) of the matrix $\boldsymbol{G}_t$ and ε is a smoothing term to avoid division by zero. This can be factorized over all parameters as:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \frac{\eta}{\sqrt{\boldsymbol{G}_t + \varepsilon}} \boldsymbol{g}_t, \tag{2.23}$$

As the learning rate is automatically adapted, there is usually no need to manually tune it and the default value of 0.01 can be leaved unchanged. The main limitation of this algorithm is the monotonic decrease of the learning rate due to the sum of squared gradients which monotonically increases as training progresses.

Adadelta (Zeiler, 2012) proposes to solve this weakness, by using a decaying average of the past gradients. The running average is then defined as:

$$E[\boldsymbol{g}^2]_{t+1} = \gamma E[\boldsymbol{g}^2]_t + (1-\gamma)\boldsymbol{g}_t^2, \tag{2.24}$$

with γ a weighting parameter. The parameter update then become:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \frac{\eta}{\sqrt{E[\boldsymbol{g}^2]_t + \varepsilon}} \boldsymbol{g}_t, \tag{2.25}$$

In order to have the update unit match the one from the parameters, an additional exponentially decaying average of the squared parameter update is introduced which I will not describe here.

The adaptive moment estimation (Adam) method (Kingma and Ba, 2015) improves on Adadelta by combining its exponentially decaying average of past squared gradients with an exponentially decaying average of past gradients like the momentum method. The authors showed that the algorithm compares favorably to other optimization methods while requiring little tuning of its hyperparameters, which makes it a good alternative in most cases.

Finally, I would like to mention a popular technique called early stopping. The method consists in monitoring the validation error and ending the training when the validation error stops improving for several consecutive steps. It proves to be an effective way of preventing overfitting of networks.

A comprehensive review of gradient descent optimization algorithms is proposed by Ruder (2016).

2.6 End-to-end models

Recently, an alternative approach to the dominant HMM-based speech recognition has attracted much attention: E2E modeling. The E2E approach proposes to use all-neural S2S[2] models to directly predict the sequence $\boldsymbol{w}^*$ (or a sub-word equivalent such as character- or phoneme-based sequence), incorporating the acoustic and language models in a single architecture. The alignment problem is also included directly into the optimization framework of those models. E2E models have the advantage of greatly simplifying the training and deployment pipelines of speech recognition systems.

One of the first breakthroughs came from the CTC loss (Graves, Fernández, et al., 2006), which allows an acoustic neural model to be trained directly on unsegmented data.

[2]E2E learning is defined as the possibility of training the model to predict the final output (transcripts in the case of speech recognition) directly from the input (speech features) in a single optimization framework. S2S models are defined as architectures that can take a variable-length sequence as input and output another variable-length output. ASR being a sequence-to-sequence problem, E2E methods are necessarily S2S. The reverse is not necessarily true. An example would be a S2S architecture using a LM for hypothesis rescoring and would thus not be E2E.

While the original technique is not E2E, it has later been extended to train models that predict grapheme sequences (Graves and Jaitly, 2014) or in conjunction with a LM based on RNNs, an architecture referred to as the RNN-transducer (Graves, 2012). More recently, the attention-based encoder-decoder model has been applied to ASR (see e.g Chorowski, Bahdanau, Cho, et al., 2014; Chan, Jaitly, et al., 2016). In this approach, the variable-length hidden representation computed by the encoder is converted at each output time step to a fixed-length vector through a weighted sum, which weights are provided by the attention mechanism. The decoder then predicts the output based on this fixed-length vector.

2.6.1 Performance

If the simplicity of the training procedure of E2E systems is attractive, they used to offer reduced performance over traditional HMM-based systems, especially so when used without an external LM. For example, the attention-based S2S model trained by Chan, Jaitly, et al. (2016) achieves results 76% relatively worse than a state-of-the art HMM-based model (reduced to -29% rel. with the use of an external LM). Using a much bigger dataset ($\sim$12,500 hours), Prabhavalkar, Rao, et al. (2017) managed to reach competitive results on a dictation task, but were still performing 13–35% worse on voice-search data. Very recently, Chiu et al. (2018) managed to reach state of the art performance on both dictation and voice search tasks (using the same 12.5K hours dataset), thanks to some of the many improvements proposed in the last years to improve S2S architectures, such as the multi-head attention. This mechanism, originally proposed by Vaswani et al. (2017), allows the decoder to attend to several locations of the encoded embedding.

While the work I just presented rely on very large dataset, E2E models can also perform well in lower resource conditions. For example, Rosenberg et al. (2017) achieved competitive results on several languages using as few as 40 hours of data, even though they failed to surpass a DNN-HMM baseline.

2.6.2 Attention-based encoder-decoder architecture

A more thorough review of the attention-based encoder-decoder architecture, which is used in chapter 5, is now proposed.

Between the different E2E approaches, the attention-based encoder-decoder architecture has been shown to give better results (Prabhavalkar, Rao, et al., 2017). While the original model (Bahdanau, Cho, et al., 2015) was proposed for neural machine translation (NMT), several ways to adapt it for speech recognition have since been proposed.

A first difference with machine translation resides in the ratio between the length of the input and output sequences: in speech recognition, the input sequence tends to be much longer than the output sequence. Chan, Jaitly, et al. (2016) proposed to use pyramidal layers to downsample the input, where the time resolution in a layer is divided by 2 by concatenating the output of every 2 units from the layer below. This reduces the number of hidden states the attention has to attend to, thus improving both the accuracy and the computational performance. Similarly, convolutional neural networks (CNNs) have been shown to be effective (Hori et al., 2017), leading to further improvement.

Another concern pertains to the global attention mechanism which is a bit too flexible for speech recognition (an essentially monotonic left-to-right process). Indeed, during training, the attention mechanism can access the whole input sequence (global property) while only a small portion of it is relevant for the prediction of next output token in speech recognition. The attention can also jump from a position to another with no constraint on the distance between the two positions and their chronology (i.e. the second position can be located before the first one). This can lead to corrupted alignments (especially for long sentences or in noisy conditions). Several strategies have been proposed to enforce a monotonic left-to-right attention better fitting the speech process. In their pioneering work, Chorowski, Bahdanau, Cho, et al. (2014) already proposed a penalty based on the distance between current and previous attention to encourage locality. They later proposed a location-aware attention mechanism (Chorowski, Bahdanau, Serdyuk, et al., 2015) where the previous attention weights are fed to the attention mechanism. Convolutional features are computed based on a learned convolutional kernel, allowing the attention mechanism to base its predictions on the previous attended location. In the same vein, Tjandra et al. (2017) proposed to predict the center of the next attended location as an increment over the last predicted center (thus enforcing monotonicity). By only considering a small window around this new center, locality is also enforced.

The local monotonic attention mechanism proposed by Tjandra et al. (2017) still relies on the full embedding from the encoder when computing the center of the attention for next output. Other approaches proposed to restrict the input to a smaller window. While this helps enforcing locality and monotonicity, it also has computational advantages as it reduces the size of the embedding over which attention computations are performed. It may also have interesting properties in the context of command recognition as it may better enable online processing of speech, where processing can start before all the utterance is acquired. One such approach has been proposed by Bahdanau, Chorowski, et al. (2016), using a fixed length window starting from the median of the previous attention weights. Alternatively,

Jaitly et al. (2016) explored the possibility to use fixed blocks of equal length. A block is discarded when the decoder produces an end-of-block symbol. In both cases though, a large window/block length has to be used to get good results, which limits the effectiveness of the approach so far.

Taking a different approach, a hybrid CTC/Attention architecture trained in a multi-task fashion has been proposed by Kim et al. (2017) (see also Watanabe, Hori, Kim, et al., 2017, for a more extended presentation). The idea there is to use the monotonous and left-to-right properties of CTC to find better alignments, which compensate for the over-flexibility of the attention-based decoder.

It has also been reported that attention-based S2S models sometime produce incomplete transcriptions. A common solution to this is to use a coverage term (Chorowski and Jaitly, 2016) which encourages the attention weights to cover the input more completely over the whole transcription process.

2.6.3 Output targets

A grapheme- or phoneme-based output is commonly used with S2S models, with phonemes having the inconvenience of requiring a pronunciation dictionary. Recently, the acoustic-to-word approach proposed to directly use words as output for S2S models (more precisely with CTC-based models). First attempts showed promising results despite the method requiring order of magnitude more data to model accurately large vocabularies (Soltau et al., 2016; Audhkhasi, Ramabhadran, et al., 2017). Attempts have been made to improve the computational efficiency (Soltau et al., 2017) and the data frugality (Audhkhasi, Kingsbury, et al., 2017) of these models.

Another interesting direction is the use of a conjunction of word and sub-word units (sometime called "word-pieces") to model OOV words. Originally proposed for NMT (Sennrich et al., 2016; Wu et al., 2016), it has been successfully applied to acoustic-to-word CTC models (Li, Ye, et al., 2018), recurrent neural network transducer (RNN-T) architecture (Rao et al., 2017), or attention-based approaches (Chan, Zhang, et al., 2017; Chiu et al., 2018; Zeyer, Irie, et al., 2018).

2.6.4 Online processing

The use of a local window when computing the attention weights (Bahdanau, Chorowski, et al., 2016; Jaitly et al., 2016) is a first step in enabling online processing with S2S architectures. Another limit of the approach lies in the bidirectional RNN layers often used

in such approaches. Bidirectional RNNs – especially bidirectional LSTMs – have been shown to be more powerful than their unidirectional equivalent. The main drawback of such layers though is the necessity to wait for the end of the sentence to perform the backward computation. This adds latency which is highly detrimental in an online pipeline. Work on windowed bidirectional RNNs (Zeyer, Schlüter, et al., 2016) could solve this issue.

2.7 Keyword spotting

In KWS, the goal is to find occurrences of one or more keywords in audio recordings. The two main applications of KWS are spoken document retrieval (SDR) and vocal command systems (sometime referred to as online or streaming KWS). Each of these applications take the problem from an opposite point of view. In SDR, the goal is to retrieve audio documents from a database based on a keyword given by the user. The audio documents are collected before the search occurs while the keyword is given at search-time only. Conversely, in vocal command systems, the list of keywords is known before-hand while the audio stream, where the keyword is to be searched for, is only available at search-time. This difference has a strong incidence on the requirements and the methodologies applied to both problems. It is common in SDR to pre-process the data to allow faster search when a query is generated. However, as the keyword is not known at this point, an open or large vocabulary should be used to not restrict the queries that can be made afterward. In vocal command systems, the audio stream has to be treated online, thus imposing strong computational and latency constraints on the keyword spotting system. Though, as the list of possible keywords is known before-hand, the system can be tuned for them in order to make it more efficient.

A field of research that is closely related to KWS (particularly the SDR task) is that of query-by-example, where the query keyword is given verbally instead of textually. The comparison with audio recordings can then be made directly in the acoustic domain. This setup is less relevant in the context of spoken commands and will not be treated here.

I will now give an overview of the different approaches devised for KWS. Contrary to a common way to classify them based on the methodology used (e.g. HMM-based versus neural network based), I will present them based on the high-level strategy employed. I will start by presenting the lattice-based search methods, move to keyword-filler models to finish with the (few) approaches that try to estimate directly a confidence score for the presence of each keyword, without resorting to a filler model. A special focus will be put on the application of these methods to spoken command interfaces as it is the topic of this thesis.

2.7.1 Lattice-based approaches

A straightforward approach to the KWS problem is simply to apply a large-vocabulary continuous speech recognition (LVCSR) system on the audio records, in order to extract a textual representation of what is said. This representation can then be search textually for the keyword. While the 1-best output of the ASR system can be used directly, N-best lattices have been shown to give better results (Szöke et al., 2005; Miller et al., 2007). Indeed, ASR systems are never perfect and will inevitably output wrong transcriptions. In such cases though, the correct words are likely to be present in the lattice with high probability. Searching the lattice then tends to produce higher recall (which is the probability of detecting keywords when they are present), with the risk of generating more false detections. A threshold can be further applied on the confidence of each hypothesis in the lattice to control such false detections.

With the recent progress of ASR systems, this approach can deliver very high performance. It also has the advantage that knowledge of the keyword(s) is not necessary to generate the lattice. This proves useful in SDR context, where the lattice can then be generated in advance and stored. Textual search over the lattices, which is very efficient, can be done at query-time, thus optimizing the whole process. The main disadvantage in that case is the dependence on the vocabulary and the grammar of the LVCSR system. Named entities for example, despite being useful keywords, are hardly manageable in such conditions.

One way to solve this issue is to work at the sub-word level. Speech recognition systems based on phonemes (Szöke et al., 2005) or other sub-word units (see e.g. Garcia and Gish, 2006, who use an unsupervisedly learned sub-word inventory) have been used to generate unconstrained lattices (vocabulary- and grammar-wise). Keywords can then be searched based on their sub-word transcription. In addition, partial matches can also be detected and the edit distance used as a confidence score (see e.g. Miller et al., 2007).

The approach used by Garcia and Gish (2006) is original from several points of view. Segment-based sub-word units (which are phone-like and syllable-like units according to the authors) are used instead of the more common phonemes or words. Moreover, it is one of the rare approaches where these units are learned unsupervisedly from the speech corpora. Only a low amount of word-level transcription (as low as 15 mins) is required to train a grapheme-to-sound model, which is used to map the keyword to its sub-word transcription. A search based on dynamic programming is then used to detect the keyword in the 1-best transcription of the audio record.

Though, as shown by Szöke et al. (2005), sub-word lattices tend to perform worse than word lattices. This can probably be attributed to the limited language constraints (in

terms of vocabulary and grammar) put on this type of models. This brings us to one of the main observation pertaining to KWS: despite the task not requiring to model non-keyword speech or non-speech signal, performance tend to increase with the capacity of the system to model them accurately. Hence, LVCSR systems which try to model the whole speech sequence tend to perform better than other techniques. This is of particular relevance in the context of streaming KWS as more powerful models are usually more demanding in terms of computations. A trade-off has then to be found between computational efficiency and the capacity to model arbitrary speech (and thus the performance).

More recently, Rosenberg et al. (2017) compared end-to-end architectures (a CTC-based model and an encoder-decoder with attention) with a more traditional DNN-HMM system in low-resource conditions. The different models where tested for both ASR and KWS tasks. In the later case, search based on word or grapheme lattices was performed (where the grapheme-based lattice was used for OOV words). Unlike ASR, poorer performance was reported for KWS with the end-to-end models, mainly due to their sharp posteriors resulting in a limited hypothesis space. Zhuang et al. (2016) also proposed an end-to-end all-neural approach, where an architecture based on LSTM units trained with the CTC loss is used to produce the phoneme lattices. Despite working at the phoneme level, and unlike previously mentioned work, this model showed promising results.

One of the main drawbacks of lattice-based search in the context of vocal command detection is the computationally intensive pipeline they require. This is not much of a problem in SDR where documents can be processed offline. It is however a big limitation when the system has to be used online. Phoneme-based (or more largely sub-word based) lattices are more tractable than word-based lattices but tend to give worse performance. Also, in both cases, transcription of keyword and non-keyword speech is treated equally. An exception to that is the work by He, Prabhavalkar, et al. (2017) which used a RNN-T model to produce phoneme- or grapheme-based output where the keyword can be searched for. In addition to specifically addressing the streaming condition, they propose an attention mechanism to bias the model toward a specific keyword at runtime. While this approach has the advantage of allowing unconstrained keyword detection, the model is trained with about $\sim$ 18000 hours of data. This illustrates another disadvantage of the lattice-based approach: while large vocabulary ASR systems produce strong results when trained on large datasets, the same level of performance is not guaranteed for smaller datasets. The keyword-filler strategy can address both issues.

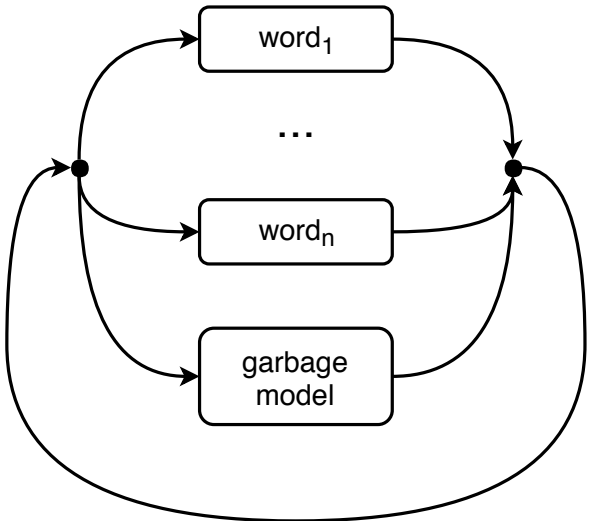

FIGURE 2.3 Generic configuration of a keyword-filler model, with the word models and the garbage model organized in a loop.

2.7.2 Keyword-filler modeling

The keyword-filler approach tries to find a middle-ground between the two extremes offered by LVCSR and phoneme-based lattice search regarding vocabulary and grammar constraints. While the keywords are modeled accurately, a garbage or filler model is used for the non-keyword speech which models it more or less accurately (illustrated in Figure 2.3). The approaches found in the literature range from a single garbage model to a CI phoneme loop with a bigram language model for the transition between the phonemes. For the keywords, more accurate CD phoneme models or word models[3] can be used. At runtime, the Viterbi algorithm is used to compute the best path and a keyword is detected when this path goes through the corresponding word model.

Comparisons of different granularity for the garbage model have been made by Bourlard, D'hoore, et al. (1994) or Manos and Zue (1997). Bourlard, D'hoore, et al. experimented with a unique garbage model, a CI phoneme loop or an intermediate model composed of 7 clusters obtained by merging similar CI models. Comparing the different methods with parameters leading to equal recognition and rejection performance, the authors found the CI phoneme loop to be stronger than the single garbage model, but weaker than the clustered models. Manos and Zue compared a LVCSR system with a CI phone-like filler composed

[3] An advantage of the small vocabulary considered in KWS is the possibility to use word models rather than phoneme models. Recent work has also explored word modeling for large vocabulary ASR (Soltau et al., 2016; Soltau et al., 2017; Audhkhasi, Ramabhadran, et al., 2017; Audhkhasi, Kingsbury, et al., 2017; Li, Ye, et al., 2018) in the E2E framework but very large datasets are required to achieve competitive performance (see subsection 2.6.3 for more details).

of 57 (segment-based) models, as well as lower levels of granularity obtained by clustering the CI models (with 18, 12 or 1 cluster). Stronger garbage models consistently give better results, at the expense of more computations. Overall, phoneme loops tend to give a good compromise between performance and computational constraints.

Similarly, Szöke et al. (2005) compared the use of a LVCSR system (which imposes strong constraints vocabulary- and grammar-wise) and a phoneme recognizer (minimal constraints) to a phoneme-based keyword-filler approach. Again, modeling extraneous speech accurately tends to improve keyword detection, and the keyword-filler approach proves to be a good compromise between the two other approaches. This is especially true for streaming applications as the set of keywords is fixed in advance.

All-neural approaches have also been used in this context, where the neural network directly predicts the probability of each frame to belong to one of the keywords or the filler model. One of the first attempts of this kind was proposed by Chen et al. (2014) using a feed-forward DNN. Specifically aimed at streaming applications, this model was shown to outperform a keyword-filler model. The simplicity of the architecture, both from the training and deployment points of view, make it very appealing. No complex decoding is required and it can be trained end-to-end. An extension to CNN-based networks (Sainath and Parada, 2015) has also been proposed. With embedded systems in mind, the trade-off between the number of parameters of the model (or the number of operations performed), and the performance is studied. Tang and Lin (2018) studied the inclusion of residual layers and dilated convolutions to further improve the performance, while Shan et al. (2018) explored the used of an attention-based architecture in the same context. While this fully neural approaches seems very competitive for keywords, they have never been applied on full sentences (as the ones we are considering in this thesis). Chen et al. (2014) proposes a simple mechanism to detect expressions composed of several words by multiplying the score of the individual words over a window. This strategy has its limits as it does not take the word order into account, or the presence of other words (such as a negation) in between (issues partially solved by Prabhavalkar, Alvarez, et al. (2015)). The use of a recurrent architecture, as done by Shan et al. (2018) may be another way to solve these issues.

2.7.3 Without explicit garbage model

The approaches I will now present all have in common that they do not use an explicit model for extraneous speech at all. The online garbage model proposed by Bourlard, D'hoore, et al. (1994) is such an approach. The score of the online garbage is computed as the average of

the N best phoneme models used to represent the keywords. This way, no additional model is required for the non-keyword speech. This approach also proved to be the most successful one (against a single garbage model, a CI phoneme loop and a compressed version of the later one with 7 phoneme clusters).

Junkawitsch et al. (1996) proposed a similar approach. Keywords are modeled with a phoneme-based GMM-HMM and decoded with a modified version of the Viterbi algorithm. Keywords are allowed to start and stop at any time step. At each time step, the decoding algorithm keeps track of a normalized score for each state, corresponding to the best path leading to this state. The normalized score relates the score of state s_j to the state with highest emission probability, similarly to the online garbage model of Bourlard, D'hoore, et al. (1994), but directly within the score. A keyword-specific threshold is used for decision, for which an optimal value can be computed analytically. Advantages of the method include low computational and storage requirements, as well as the possibility to run the algorithm time-synchronously. However, the performance of the system is not compared against any other system making it hard to evaluate its performance. Also, the algorithm is susceptible to changes in the acoustic conditions (e.g. noise).

Finally, Keshet et al. (2009) proposed a discriminative method based on support vector machines (SVMs). The acoustic features corresponding to the sentences to test and the target keyword are mapped into the same vectorial space, where a SVM is used to separate sentences with and without the keyword. The method outperforms a conventional HMM-based approach, but relies on manually designed feature functions.

2.8 Evaluation

2.8.1 WER

The WER is the most common evaluation metric for speech recognition systems. To compute it, the hypothesized transcription is aligned with the reference transcription through dynamic programming techniques. As a result, several indicators can be computed:

- S: the number of words that were substituted with a different one in the hypothesized transcription
- D: the number of words that were deleted
- I: the number of words that were inserted

TABLE 2.1 Confusion matrix for keyword detection.

		True condition	
		Keyword present	**Keyword absent**
Predicted condition	**Keyword detected**	True positive	False positive
	Keyword not detected	False negative	True negative

- C: the number of words that were correctly predicted
- N: the total number of words in the reference ($N = S + D + C$)

The WER can then be computed as:

$$WER = \frac{S + D + I}{N} \tag{2.26}$$

It is also possible to define an accuracy score $WAcc$ as:

$$WAcc = 1 - WER = \frac{N - S - D - I}{N} = \frac{C - I}{N}. \tag{2.27}$$

2.8.2 KWS

When dealing with KWS systems, it is more common to report the performance with a confusion matrix, in terms of the predicted condition (positive if a keyword is detected, negative otherwise) and the truthness of this prediction. Table 2.1 represents the 4 possible outcomes.

Often, some hyperparameter is available which allows to trade between true and false positive rates. By increasing the probability of detecting a keyword, we are less likely to miss one but also more likely to detect one when it is not present. The evolution of the true positive rate as one increase the false negative rate corresponds to the receiver operating characteristic (ROC) curve. A good KWS tries to maximize the area under this curve.

Another way to evaluate a KWS system is through the precision and recall scores, defined as:

$$precision = \frac{\sum true\ positive}{\sum keyword\ detected} = \frac{\sum true\ positive}{\sum (true\ positive + false\ positive)} \tag{2.28}$$

$$recall = \frac{\sum true\ positive}{\sum keyword\ present} = \frac{\sum true\ positive}{\sum (true\ positive + false\ negative)} \tag{2.29}$$

Those two values can be combined in a single score called the F_1 score and defined by:

$$F_1\ score = \frac{2}{\frac{1}{recall} + \frac{1}{precision}} \tag{2.30}$$

2.9 Proposed approach

Unlike what is usually done for commercial products incorporating vocal command recognition, this thesis propose to explore direct detection of commands without the use of a wake-up word. However, KWS approaches to streaming application focus on single word commands and are hardly applicable to direct detection of fully-fledged sentences. For the iCubrec pipeline, I thus propose the use of a VAD system as a first step. This should allow to filter most of the silence and reduce the computational load of the system in large proportions. The audio segments extracted by the VAD system can then be processed by a more classical system based on HMMs, thus allowing to reuse the large number of techniques available in that context (such as domain adaptation), or a more original E2E system. In both cases, the model will have to deal with the presence of non-command sentences which will require the use of a garbage model.

Another important aspect of a spoken command recognition system is usually the problem of command understanding. As this is not the subject of the thesis, I simply relied on a one-to-one mapping between each command and a corresponding action (said otherwise, the language understanding module is reduced to a look up table).

Chapter 3

Resources

3.1 Toolkits

The two main toolkits available for speech recognition are the Hidden Markov Model Toolkit (HTK) (Young, Evermann, et al., 2015) and Kaldi (Povey, Ghoshal, et al., 2011). While Kaldi tends to be favored by the speech community lately, thanks to its active community and the availability of many recent techniques, HTK has the advantage of being better documented and more accessible for newcomers. For these reasons, I initially used HTK for my experiments. The work I did on domain robustness and the first version of iCubrec rely on this framework. More recently, I moved to Kaldi as it offers more flexibility. The CNN-HMM baseline in chapter 5 and the new version of iCubrec where both implemented with Kaldi. I will quickly describe both frameworks. I will also present TensorFlow (Abadi et al., 2016) deep-learning library, used to train the DNN-based acoustic models in both cases. While HTK required the development of transfer tools to convert the neural networks trained with Tensorflow into its own format, Kaldi can natively take as input posteriors computed with an external model. Finally, the End-to-End Speech Processing Toolkit (ESPnet)[1] will be presented, which was used to train sequence-to-sequence models.

3.1.1 The Hidden Markov Model Toolkit

HTK is a toolkit for building and manipulating HMMs, which is mainly used for speech recognition. Currently maintained by the Cambridge University Engineering Department (CUED), the code is freely available for academic research. It consists of a set of libraries and modules that provide facilities for all the tasks necessary to build a speech recognition

[1] https://github.com/espnet/espnet

system, such as feature extraction, HMMs estimation and testing or result analysis. The output probabilities of the HMM model can be estimated with GMMs or DNNs, for which native support has been recently added (as of version 3.5) (Zhang and Woodland, 2015).

Neural network functionalities remain limited though: only feed-forward networks can be used, with a limited set of activation functions or training options. In order to alleviate this constraints, I alternatively trained networks with an external deep learning framework (TensorFlow in this case) and converted the trained models to HTK format for decoding. This can be done easily with the appropriate code and allows to use more complex optimization procedures, even though we are still limited to the architectures recognized by HTK.

A strong advantage of HTK is its accompanying book (Young, Evermann, et al., 2015) which provides an extensive documentation to the toolkit.

3.1.2 Kaldi

Kaldi, named after the discoverer of the coffee plant, is a modern speech recognition toolkit freely available under the Apache License. Its development started in 2009 and was based on HTK, but quickly became an independent product. A particular effort is made to propose a flexible and extensible set of functions. For instance, and contrary to HTK, Kaldi can easily be interfaced with external deep learning libraries. It also provides complete recipes for the main speech datasets. Mainly intended for researchers, its documentation is not always accessible to non-expert users.

3.1.3 TensorFlow

TensorFlow is not a speech recognition framework per se but can be used to train DNN-based acoustic models. Mainly developed by Google, it is an open source software library (released under the Apache 2.0 license) for high performance numerical computation, mainly used for machine learning and deep learning applications. Python is the main programming language supported by TensorFlow and the one supporting the widest range of features. An application programming interface (API) in C language is also available with less functionalities. Tensorflow originally represents the flow of computations through a graph, which has to be compiled before it can be applied to any data. More recently, an imperative programming environment called "eager mode" has also been added which allows immediate evaluation of operations (officially announced in version 1.7).

TensorFlow was used to train all the DNN-based acoustic models presented in this thesis, in conjunction with either HTK or Kaldi.

3.1.4 End-to-End Speech Processing Toolkit

The ESPnet toolkit was originally developed for experiments on hybrid CTC/Attention end-to-end models (Kim et al., 2017; Hori et al., 2017; Watanabe, Hori, Kim, et al., 2017). It is now freely available and provides a complete setup for speech recognition. Both pytorch[2] and chainer[3] deep-learning libraries can be adopted. Kaldi's data preprocessing and scoring facilities are used, from which the recipe style was also borrowed.

Technically, the toolkit proposes different kinds of encoders (based on CNNs or pyramidal bilateral long short-term memory (BiLSTM)) and attention mechanisms (dot product, location-aware or multi-head, see section 2.6). It also supports the use of a language model for rescoring (based on RNNs or LSTMs) and provides recipes for many well-known datasets (e.g. Wall Street Journal (WSJ), Switchboard, CHiME 4 and 5, Librispeech, TED, AMI, Voxforge).

3.2 Datasets

The different datasets used in the thesis will be presented in following subsections. When presenting baseline systems, I will limit myself to DNN-based acoustic models as it is the focus of this thesis. In all the cases, the alignments are obtained from a GMM-HMM system, which is also used to train the HMM and provide the triphones clustering (TIMIT excepted as we use monophone targets).

I will start with the TIMIT dataset (Garofolo, Lamel, et al., 1993) used in the experiments on domain adaptation. I will then present the WSJ dataset (Garofolo, Graff, et al., 1993). WSJ is not used on its own but is the starting point of CHiME4 (Vincent et al., 2017) which was used as a source corpus for experiments on domain robustness with the VoCub dataset in chapter 6. The Speech Commands (SC) dataset (Warden, 2018) is used in the experiments on few-shot learning for S2S models in chapter 5. Finally, I will introduce the VoCub dataset (Higy, Mereta, et al., 2018) that I gathered in the scope of this thesis to address the problem of command recognition for robotics.

[2] https://pytorch.org/
[3] https://chainer.org

3.2.1 TIMIT

Dataset description

TIMIT (Garofolo, Lamel, et al., 1993) is a dataset of clean read speech that was commissioned by the Defense Advanced Research Projects Agency (DARPA). It was recorded at Texas Instruments (TI) and transcribed and Massachusetts Institute of Technology (MIT), hence its name.

It is composed of recordings from 630 american English speakers from 8 major dialects and both genders, who read 10 sentences each. From these 10 sentences, 2 are identical for all speakers (referred to as the `SA` sentences in the corpus), 5 of them are read by 7 different speakers (referred to as the `SX` sentences) and the remaining 3 sentences are unique for each speaker (referred to as the `SI` sentences). The dataset has been designed to be phonetically balanced and provides both phonetic and lexical alignments.

Validation and coreTest sets

The dataset provides a default split of speakers in training and test sets. However, some of the `SX` sentences read by speakers in the test set are shared with speakers from the training set. To avoid that, a core test set is also specified in the documentation. The speakers from this core test set read sentences different from the ones used in the training set. In all the experiments with TIMIT, this core test set was used as the actual test set. The remaining set of speakers, which are part of the original test set but share some sentences with the training speakers, form the validation set.

Evaluation

The TIMIT corpus being meant for acoustic-phonetic studies, it is standard to evaluate models trained on it in term of phone error rate (PER) and not WER, as is usually done with other datasets. In the same way, a phone bigram LM is used. I followed this standard procedure in all experiments reported in this thesis.

3.2.2 Wall Street Journal

WSJ (Garofolo, Graff, et al., 1993) is also a DARPA dataset that was built to support research on LVCSR systems and consists of clean read speech. The utterances are selected from a corpus of Wall Street Journal news and are divided into two subsets recorded in different phases, WSJ0 and WSJ1. WSJ0 is composed of the 7138 utterance and adding WSJ1 brings

TABLE 3.1 Summary of the number of sentences and speakers in CHiME4 dataset.

Subset	Real data		Simulated data	
	# of speakers	# of sentences	# of speakers	# of sentences
tr	4	1600	83	7318
dt	4	1640	4	1640
et	4	1320	4	1320

this number to 36515 sentences. Some spontaneous dictation is included in addition to the read speech.

Training, validation and test sets are provided for speaker-independent (SI) and speaker-dependent (SD) models, with vocabularies of 5k or 20k words. Bigram language models are provided for both vocabulary sizes, with or without verbalized punctuation and for open or closed vocabulary sets. Recordings from two microphones are available, a close-talking Sennheiser HMD414 and a second microphone which may vary.

3.2.3 CHiME4

The 4th Computational Hearing in Multisource Environments (CHiME) challenge[4] (Vincent et al., 2017) addressed distant speech recognition. The accompanying read speech corpus is based on WSJ setup, more precisely the SI 5k vocabulary ("no verbal punctuation" – NVP) subset of WSJ0. Unlike this later dataset, recordings in CHiME4 (real and simulated) correspond to distant speech in noisy conditions. Four kinds of environments are considered: bus, cafeteria, pedestrian area, street junction.

Real recordings for training (`tr`), validation (`dt`) and test (`et`) sets correspond to data from 4 speakers each. Simulated data were generated by artificially mixing clean speech with pre-recorded noise. For the training set, the original 7138 utterances from WSJ0 are used. For validation and test sets, utterances recorded in a booth are used to generate simulated data. Table 3.1 summarizes the number of speakers and sentences in each subset.

For all recordings, 6 channels are available, corresponding to 6 different microphones present on the tablet used to prompt the sentences (5 in the front and 1 in the back of the device). For real data, an additional channel is available corresponding to a close microphone. The challenge was then divided into 3 tracks depending on the number of microphones available at test time – 6 channels, 2 channels or 1 channel. For the two last tracks, the

[4] http://spandh.dcs.shef.ac.uk/chime_challenge/chime2016/data.html

microphones used for each utterance in the validation and test set were randomly sampled from the 5 front channels.

3.2.4 Speech Commands

Description of the dataset

Google's SC dataset (Warden, 2018) is a corpus of spoken words designed to meet the specific requirements of KWS. The dataset tries to provide freely accessible data to train and evaluate such systems, with the aim of making vocal interfaces available to a wider audience of researchers and makers, a philosophy that corresponds to my objective of making command recognition technology more accessible. Unlike VoCub dataset, the corpus focuses on keywords but has some nice properties: it provides many examples per keyword (with a mean of ∼3000 examples per classes), it covers a wide variety of accents and data was collected in realistic conditions. This properties made it a good candidate for my experiments on sequence-to-sequence and few-shot learning (chapter 5).

The first version of the dataset was composed of 30 keywords with ∼65,000 examples overall. A second version of the dataset was released recently, which brings these numbers to ∼105,800 recordings for 35 different keywords, pronounced by 2618 speakers. Each record has a fixed duration of 1 second and contains an isolated word, so as to resemble a trigger task, where a wake-up word is used to activate speech recognition on a device. It is composed of a core of 24 common words, including the 10 digits and commands that could be useful for Internet of Things (IoT) or robotics (e.g. *yes*, *no*, *on*, *off*). The 11 remaining words are mainly meant to populate an `_unknown_` category, corresponding to OOV speech. This group tries to cover as many phonemes as possible and also contains some words with a pronunciation close to that of some core keywords (e.g, *tree* which is close to *three*).

The precise task suggested by Warden (2018), which was also the goal of a Kaggle challenge[5], corresponds to the classification of 10 of the keywords (out of the 35 available). Two additional classes are considered: (i) a `_silence_` category corresponding to records free of speech, and (ii) an `_unknown_` category for records containing speech that is none of the keywords. Several minute-long records of background noise are bundled with the dataset and can be used to generate examples for the `_silence_` category, while the `_unknown_` category is populated with the remaining 25 keywords. See Table 3.2 for the list of words by category (target keywords vs. unknown words).

[5] https://www.kaggle.com/c/tensorflow-speech-recognition-challenge

TABLE 3.2 List of the words present in the SC dataset by category. Words with an asterisk are new in version 2 of the dataset.

Set	Words
target keywords	down, go, left, no, off, on, right, stop, up, yes
unknown words	backward*, bed, bird, cat, dog, eight, five, follow*, forward*, four, happy, house, learn*, marvin, nine, one, seven, sheila, six, three, tree, two, visual*, wow, zero

3.2.5 VoCub

Recording a dataset has two main advantages: (i) it allows to easily test the recognition system and reliably estimate its performance in real conditions, and (ii) it can be used to adapt the system in order to reduce the training-testing mismatch problem (see subsection 4.1.1 for the definition). For this reasons, I gathered a dataset corresponding to a real-usage scenario with the robot that will be used as a test application. This resulted in the VoCub dataset[6] (Higy, Mereta, et al., 2018) that will be presented in the following.

Target application

As the target application on the robot, I chose the interactive object learning (IOL) demo[7]. The purpose of this demo was to demonstrate visual object learning and classification on iCub. The robot is presented with new objects together with a verbal label and learns the object's visual appearance. It can then recognize the different items and perform simple actions such as grasping or pointing to them.

For the dataset, 8 different objects have been selected (illustrated in Figure 3.1), leading to 103 unique english commands (see Table 3.3 for some examples or Appendix A for the complete grammar).

Acquisition process

The commands were recorded with a tablet, in order to have a more generic setup that does not require the robot to collect training data. Recordings were done in the same conditions as normal experiments with the robot, that is in the room where experiments with iCub are

[6]The dataset can be freely downloaded from the accompanying website at https://robotology.github.io/natural-speech/vocub.

[7]A demonstration video can be found at https://youtu.be/ghUFweqm7W8, where a pre-existing command recognition system based on Microsoft SDK is used.

FIGURE 3.1 The 8 objects selected for the VoCub dataset. From top to bottom and left to right: bottle, ladybug, turtle, toy, box, octopus, lego, car.

TABLE 3.3 10 examples of the commands used in the VoCub dataset

I will teach you a new object.
This is an octopus.
What is this?
Let me show you how to reach the car with your left arm.
Let me show you how to reach the turtle with your right arm.
There you go.
Grasp the ladybug.
Where is the car?
No, here it is.
See you soon.

usually carried over and with the same environmental noise (noisy servers and computers running, possible concurrent speakers in the background).

The data collection was split in two phases for each speaker, a non-static (`cond` = 1) and a static condition (`cond` = 2), with an equal number of recorded utterances for each condition. In the static condition, the speaker sat in front of two screens where the sentences to read were displayed. In the non-static condition, the commands were provided to the subject verbally through a speech synthesis system, and the subject had to repeat them while performing a secondary manual task[8]. This secondary task was designed to be simple enough to not impede the utterance repetition task, while requiring people to move around the robot. The distance between the speaker and the microphone in this last condition ranges from 50 cm to 3 m.

An additional set of sentences was collected for the testing group (same structure but different vocabulary) to test the recognition system for new commands not seen during training. The sentences consist of 20 new commands, pronounced by each speaker of the test set twice: once in non-static condition (`cond` = 3) and once in static condition (`cond` = 4). See section A.2 for the extended grammar, including the new sentences.

Dataset

The recordings consist of the 103 unique commands, which are composed of 62 different words. In total, 29 speakers were recorded, 16 males and 13 females, all of whom are non-native american/british English speakers. I finally obtained 118 recordings from each speaker: of the 103 unique commands, 88 were recorded once, and 15 twice (corresponding to sentences containing rare words, see Table A.1 in appendix). This resulted in about 2.5 hours of recording in total.

A split of the speakers into training, validation and test sets is proposed with 21, 4 and 4 speakers per set respectively. The files are organized with the following convention `<setid>/<spkrid>/<spkrid>_<cond>_<recid>.wav`, where:

- `<setid>` identifies the set: `tr` for training, `dt` for validation and `et` for testing.
- `<spkrid>` identifies the speaker: from `001` to `021` for training, `101` to `104` for validation and `201` to `204` for testing.
- `<cond>` identifies the condition (see above).
- `<recid>` identifies the record within the condition (starting from `0` and increasing).

[8] The acquisition process is better illustrated by this video: https://youtu.be/N-rrNQ0gnRY.

TABLE 3.4 Hyperparameters used to train the acoustic DNN-HMM baseline on VoCub.

Parameter	Value
Input size	440
Output size	681
Number of hidden layers	4
Number of units per hidden layer	2000
Mini-batch size	200
Momentum	0.9
Initial learning rate	0.01
Decay rate	0.95
L2-norm regularization weight	0.001
Number of epochs	30
Number of epochs for early stopping	3

Baseline

The baseline DNN-HMM model I will now present has been trained by Leonardo Badino.

The acoustic model was trained using linear discriminant analysis (LDA)-maximum likelihood linear transform (MLLT) features, extracted with Kaldi over windows of 25 ms with a stride of 10 ms. To obtain them, 13 MFCCs are first extracted with first and second derivatives. A context of 3 frames from each side is then used, resulting in a vector of dimension 234, which is reduced to 40 values through LDA. A diagonalizing transform (MLLT) is finally estimated and applied.

The DNN model, is trained through Tensorflow to predict senone probabilities. It uses the LDA-MLLT features from 11 frames (5 from each side) as input and has an output size of 681. It is composed of 4 hidden layers of 2000 units with ReLU activation function for intermediate layers and softmax function for the final one. It is trained through cross-entropy, with mini-batch SGD. A decaying learning rate with momentum is applied, as well as l_2-norm regularization. The network is trained for a maximum of 30 epochs and early stopping is applied if the performance on the validation set does not improve for 3 consecutive epochs. Table 3.4 summarizes the values of the different hyperparameters.

Decoding and evaluation are done with Kaldi, using a strict grammar (`strict` condition hereafter) allowing only the commands present in the dataset. As can be seen from Table 3.5, despite the small size of the dataset, very good results can be achieved. We reach an accuracy of 92% on the test set without the use of any advanced technique. While this is still quite low for practical use (about 1 sentence out of 10 is mis-recognized), it is promising given the size

TABLE 3.5 Evaluation of the baseline system on VoCub validation and test sets. FER of the DNN, WER and SER (or command accuracy) are given for the test set with more or less restrictive grammars

Grammar	Validation set			Test set		
	FER (%)	WER (%)	SER (%)	FER (%)	WER (%)	SER (%)
`strict`	48.4	5.62	13.56	48.6	4.14	7.64
`loop`	48.4	28.64	63.35	48.6	19.18	44.59

of the training set. To further evaluate the performance of the acoustic model and the impact of the strong grammar, I also report results where the grammar is replaced with a word loop (referred to as `loop` condition), which allows any combination of the 62 words from the dataset. Simply relaxing the syntactic constraints of the grammar results in an important deterioration of the performance from 7.64% of sentence error rate (SER) to 44.59%. This suggests that the acoustic model is not very robust and the grammar plays an important role in the good performance we observed. This motivated my work on domain adaptation which explored new strategies to improve the acoustic model leveraging on available datasets.

Table 3.6 further analyze the performance of the baseline system on the extended test set. I report performance on the original sentences (conditions 1 and 2) and the new ones (conditions 3 and 4) separately. The grammar is also extended to include the new sentences (see section A.2 for the definition of the extended grammar). Performance on the original sentences slightly improves with this new grammar. The performance on the new sentences is very poor though, with only 12% of accuracy, probably due to a poor representation of the senones present in these new sentences and not in the original ones. It can also be that the model overfitted to the original sentences. This shows another limitation of the model trained on VoCub dataset only. Again, domain adaptation may help mitigate the issue to some extent, as the source dataset can provide a better coverage of the senones. However, because of the mismatch between the source and target domains, performance may still degrade significantly. The work on few-shot learning in chapter 5 is an attempt to compensate for that, by exploring how the set of commands can be extended with only few examples for each new command.

TABLE 3.6 Evaluation of the baseline system on VoCub extended test set. WER and SER (or command accuracy) are given for conditions 1 and 2 on one side and 3 and 4 on the other side. The grammar is also extended to contain the new sentences introduced in conditions 3 and 4.

Conditions	**WER (%)**	**SER (%)**
1 and 2	3.47	7.22
3 and 4	66.08	88.05

Chapter 4

Robustness against sources of variability

As mentioned previously (chapter 2), the goal of speech recognition is to find a mapping from the input acoustic signal to the sequence of words that generated it. However, many sources of variability affect the target signal as it proceeds from the speaker to the listener. A first group of factors is related to the speaker and influence the production process itself: gender and age, accent, speaking rate or idiosyncrasies play a role in speech generation and influence what is emitted. The signal then suffers distortions while it proceeds from the speaker to the listener: the noise produced by other sources get mixed with the original signal and alter it. Also, the physical environment impacts the signal through reverberation, as the sound is reflected by the surfaces of objects present in the environment. Finally, the characteristics of the listener (be it a human or an electronic device) affect the way the signal is perceived. In the case of ASR, the difference between microphones is referred to as channel variability.

How to obtain acoustic models that are robust against these sources of variability is a long-standing research topic. In close-microphone scenarios, speaker variability is the main issue. Though, as the performance of speech recognition systems improve, they tend to be used in more adverse acoustic environments and noise robustness is taking more and more importance. This is even more the case as we also start considering distant microphone conditions: the longer the distance between the microphone and the speaker, the stronger is the impact of the noise on the recorded signal. Nonetheless, despite these two main sources of variability (speaker and noise) having their own specificities, many techniques can be applied to both factors indifferently.

The following section will present the main solutions that have been developed over the years to tackle speaker and domain variability.

4.1 Problem definition and solutions

4.1.1 The training-testing mismatch problem

Acoustic models can learn to deal with most sources of variability in the acoustic signal given adequate training data. However, in practice, they are trained on a finite set of records that can only cover those sources partially. Thus, they will likely fail to extrapolate to test samples corresponding to conditions that are substantially different from the training ones. As the discrepancy between training and testing (or runtime) conditions increases, the performance can quickly degrade. In the literature, this is often referred to as the training-testing mismatch problem. One way to alleviate the issue is to resort to multi-condition training, where the training data covers different conditions. This prevent the model from overfitting to one specific context and helps it generalize better to unseen situations.

4.1.2 Domain independent or domain dependent models

In face of the difficulty in covering all possible domains and the deterioration of performance that can result from mismatched training and testing conditions, there are two complementary stances one can take. Working on domain independent models, one can try to improve the invariance of the representation learned by the acoustic model, so that it is less dependent on the conditions in which the training data has been recorded. Alternatively, one can avoid the problem altogether by using models trained on data matched to the testing conditions only. Indeed, such domain dependent models are ensured to perform at least as well as a domain independent model given enough matched training data. However, they have several limitations. They require to train and store a different model for each domain considered. Also, for each new model to train, a big dataset is required (especially so with the recent deep learning models) and gathering it can be an expensive process.

Hence, if a domain dependent approach results in improved performance, it is traded off against the greater flexibility of domain independent models. Depending on the application's requirements, one or the other may be favored.

4.1.3 Domain adaptation

As we have just seen, domain dependent models give the best performance but require large amount of training data covering the testing conditions. When the resources available for the target domain are scarce, an alternative approach is applicable: domain adaptation. This

approach leverages a larger speech corpus recorded in different conditions (source domain) to train a model matched to the target context for which only low resources are available. An example could be a model trained on adult speech (source domain) and adapted to child speech (target domain). This time, the trade-off is between the performance and the quantity of data available.

A key aspect of domain adaptation is the compactness of the representation that needs to be learned as the limited test-matched data do not allow the proper estimation of a large set of parameters.

Another distinction is to be made between supervised methods, that require labeled data, and unsupervised approaches, that can leverage on non-labeled utterances.

4.1.4 Gaussian mixture models versus deep neural networks

The traditional GMM-HMM approach is known to be particularly bad at generalizing to unseen conditions. DNN-based acoustic models, on the other hand, have been proved to generalize better to new conditions. Seltzer et al. (2013), while investigating the noise robustness of neural networks, showed that a DNN trained without explicit noise compensation can match its state-of-the-art GMM counterpart on Aurora-4 dataset. Similarly, Liao (2013) has demonstrated that DNNs can adapt better to new speakers than GMMs.

Yu, Seltzer, et al. (2013) have further analyzed DNNs internal representation to understand their remarkable robustness to sources of variability in the input space. They have found that the invariance of neural networks is due to the robustness of their internal representation which allow them to interpolate well around training samples. They also showed that the model's invariance increases with its depth. However, despite these nice properties, neural networks may still fail to extrapolate when the data mismatch increases. This is in line with the work by Seltzer et al. (2013), who demonstrated that noise robustness can be further improved, e.g. by incorporating information about the environment (the noise-aware training strategy described in subsection 4.2.1). Hence, it is useful to try compensating explicitly for unseen variabilities.

4.2 Review of the literature

In this section, I will mainly present methods that apply to DNNs as it is the focus of this thesis. GMM-based approaches will only be presented when necessary.

4.2.1 Improving model invariance

I will first present strategies that try to improve the invariance of domain independent models.

A simple yet effective approach, already in use with GMM-HMM models, is the *multi-condition training* strategy, which consists in exposing the model to the largest set of domains possible. This allows the model to learn regularities from the data that are less dependent on a specific condition and generalize better to new ones. An example of its use for noise robustness can be found in Seltzer et al. (2013).

A more explicit way to enhance the invariance of the internal representation of neural networks has been proposed for speech by Shinohara (2016). The authors explored the use of *adversarial multi-task learning (MTL)* techniques, where the classical senone classification task is trained adversarially to a domain (noise) classifier, that is, the error from the noise classification task is subtracted when estimating the parameters of the main network, so as to reduce the noise discriminability. Saon, Kurata, et al. (2017) applied the same approach to speaker variability by training a speaker classifier in parallel to the main network.

Finally, *domain-aware* approaches propose to enrich the input to the acoustic model with information about the domain , with the idea that the neural network can relate these additional features to the variability in the data and better deal with it. This has been proved effective for both noises (Seltzer et al., 2013) and speakers (Saon, Soltau, et al., 2013; Senior and Lopez-Moreno, 2014).

As shown by Yu and Deng (2015, section 11.5.2, equations 11.19 and 11.20), domain-aware training (or equivalently speaker-aware training) is equivalent to using a domain-specific bias that is implicitly learned from the domain information. Learning the relation between the domain information and the bias allows to generalize to new domains.

A popular way to represent speaker information is to use i-vectors, originally proposed for speaker verification (Dehak et al., 2011). I-vectors try to capture information about the sources of variability in the acoustic signal (not only speaker but also channel or noise) in a low-dimensional fixed-length representation. They can be computed at the utterance (Senior and Lopez-Moreno, 2014) or speaker (Saon, Soltau, et al., 2013) level.

Going one step further, Cui et al. (2017) explored the possibility to use i-vectors to adapt the network at different levels. Following a meta-learning approach, they train a control network that will estimate scaling and bias weights that are applied after each hidden layer, instead of the bias of the first layer only.

While the domain information can be computed as a separate step, it is also possible to incorporate the domain feature extraction process into the main network. Qian et al. (2016) proposed a model that computes senones posterior probability while jointly learning

to extract speaker, phone and environment factors. These factors are then fed back to the main task to improve invariance.

4.2.2 Feature adaptation

The techniques I just presented investigate how the acoustic model can be made more invariant to the variability in the input features. Alternatively, it is possible to normalize the input before feeding it to the acoustic model.

The most popular feature-based approaches to speaker adaptation with GMM-HMM systems were vocal tract length normalization (VTLN) (Eide and Gish, 1996) and feature-space maximum likelihood linear regression (fMLLR) (Gales et al., 1998). Seide et al. (2011) studied the usefulness of these features with the newly introduced DNN-based approach and found that most of the gain from VTLN is subsumed: the error reduction goes from 9% when VTLN is used with GMMs to 2% with DNNs. The use of fMLLR transforms on the other hand is still effective with 5% of error reduction regardless of the type of model. The main limitation of the later approach is the need of a GMM to estimate the features.

More recently, Mimura et al. (2017) proposed to use generative adverserial networks (GANs) to perform cross-domain adaptation with non-parallel corporas. In that case, instead of trying to remove sources of variability from the input, they learn a mapping from source to target domains (and vice-versa), giving the possibility to map all input data to the same domain and learn a single domain-dependent model. They applied their methodology to noise and speaking style adaptation.

4.2.3 Model adaptation

Model adaptation offers a compromise between the good performance of a domain-dependent model and the difficulty to gather enough data to train one. Leveraging on well trained systems from one or more source domains with abundant data, this approach propose to learn a model better matched to the target domain with a fraction of the data needed to train a domain-dependent one. Again, this approach also applies to speaker variability.

The simplest strategy one can imagine consists in initializing the network with a pre-trained model and train it for a few epochs on the target domain or speaker, updating all the weights (see e.g. Gemello et al., 2007; Liao, 2013; Saon, Soltau, et al., 2013). Given enough data, this strategy can work well but will easily overfit if the dataset is too small. It is important then to reduce the number of adapted parameters. Several adaptation techniques

have been proposed over the years which offer a compact representation that can be estimated even from a small amount of data.

A popular strategy is to add a linear layer to the network. The weights of the linear layer are then learned using the data from the target domain. Depending on the position at which the layer is placed in the network, it will be referred to as a linear input network (LIN) (Abrash et al., 1995; Neto et al., 1995), a linear hidden network (LHN) (Gemello et al., 2007) or a linear output network (LON) (Li and Sim, 2010). The LIN approach is close to fMLLR in spirit as it offers a way to normalize the input. The best position in the network at which the layer should be placed seems to be highly dependent on the task.

Rather than adding a linear layer, Swietojanski and Renals (2014) proposed to adapt the amplitude of hidden units in an unsupervised fashion (using first-pass alignments), a technique called the learning hidden units contribution (LHUC) method. This method showed an improvement of 8-15% on a corpus of publicly available TED talks and is compatible with fMLLR.

Finally, an alternative to the addition of new parameters is the use of regularization. A famous method is to force the output distribution of the domain-dependent model to stay close to the output distribution of the domain independent network, using e.g. the Kullback–Leibler divergence (KLD) (Yu, Yao, et al., 2013). See also Falavigna et al. (2017) for an example of KLD adaptation applied to CHIME-3.

4.2.4 Speaker adaptive training

Speaker adaptive training (SAT) was introduced by Anastasakos et al. (1996) for speaker adaptation with GMM-HMM but the idea is quite generic and can easily be transposed to other sources of variability or methodologies. The main observation made by the authors is that a SI acoustic model has to model both the phonetic and speaker variability. While model adaptation techniques such as LIN try to compensate for the speaker variability only during the adaptation step, Anastasakos et al. propose to already decouple the two sources of variability at training time. The part of the model learning the phonetic regularities should thus be more independent of speakers and easier to adapt in subsequent step.

Some of the techniques presented here above can easily be modified to integrate domain or speaker adaptation at training time. For example, this has been the case for the family of linear networks (Ochiai, Matsuda, Lu, et al., 2014; Ochiai, Matsuda, Watanabe, et al., 2015) or the LHUC approach (Swietojanski and Renals, 2016).

4.3 Distillation

Another line of research that is related to the methodology employed here is knowledge distillation (Hinton, Vinyals, et al., 2015). Distillation was not specifically designed for model adaptation but can be used in this context as we will see.

4.3.1 Seminal work

Firstly coined by Hinton, Vinyals, et al. (2015), knowledge distillation stems from work on model compression (Bucilă et al., 2006). The original goal was to compact an ensemble of models of various types into a simpler model that would achieve performance close to the ensemble for a fraction of the computational cost. To do that, the complex ensemble (also referred to as the teacher) is used to label data (real or simulated) and the resulting labels are used to teach a simpler neural network based model, also called the student. Neural networks being universal function approximators, they should in principle be able to approximate any function given enough data and parameters. They are thus ideal candidates to learn the function modeled by the ensemble and represent it in a more compact way. The use of artificially generated data is key in the work from Bucilă et al. as it allows to transfer the knowledge of the ensemble (which is used to label the data) to the student network. This additional source of information about the model learned by the ensemble guides the student model to a better optimum. The application of model compression to speech recognition was first proposed by Li, Zhao, et al. (2014), where real unlabelled data is used instead of artificial data to transfer information to the student.

Hinton, Vinyals, et al. (2015) also introduced a temperature parameter T to control the smoothness of the teachers' softmax output layer. The i^{th} output of the network, $\tilde{\mathbf{y}}_i$, is then defined as:

$$\tilde{\mathbf{y}}_i = \frac{exp(\mathbf{z}_i/T)}{\sum_j exp(\mathbf{z}_j/T)}, \tag{4.1}$$

which reduces to the standard softmax function for $T = 1$. The loss of the student network is in turn defined as:

$$L_S(\boldsymbol{x}_i, \boldsymbol{y}_i) = (1-\lambda)L_{CE}(\boldsymbol{y}_i, f_S(\boldsymbol{x}_i, \boldsymbol{\theta})) + \lambda T^2 L_{CE}(\tilde{\mathbf{y}}_i, f'_S(\boldsymbol{x}_i, \boldsymbol{\theta}, T)) + R(\boldsymbol{\theta}), \tag{4.2}$$

where L_{CE} is the cross entropy loss, $\boldsymbol{x}_i$, $\boldsymbol{y}_i$ and $\tilde{\mathbf{y}}_i$ are the i^{th} input vector, one-hot label and soft label from the teacher respectively, f_S and f'_S are the student model without and with temperature in the final softmax, $\boldsymbol{\theta}$ is the set of parameters of the model and R is a

regularization term. As advised by Hinton, Vinyals, et al., the loss on the soft targets is scaled by T^2 to keep the relative contribution of soft and hard labels to the gradient similar as the temperature is changed.

The motivation behind distillation is that, in the multi-class classification task, the knowledge from the teacher can also be conveyed through the probability given to the non-target categories. This is to contrast with the one-hot label often used to train neural networks. By raising the temperature, the output of the softmax is smoothed and more of this knowledge is retained, providing more information to the student. Hinton, Vinyals, et al. also prove that, in the extreme case where an infinite temperature is used, softmax with temperature reduces to the logits (which are sometime used instead of the output to distill knowledge).

4.3.2 Generalized distillation

Lopez-Paz et al. (2015) further proposed a generalized distillation framework which combines the distillation strategy from Hinton, Vinyals, et al. with the privileged knowledge paradigm from Vapnik and Vashist (2009). This later work explores the use of some privileged information $\boldsymbol{x}^*$ that is available at training time only. An example related to speech recognition would be the availability of speaker information for each utterance of the training dataset but not at test time. Lopez-Paz et al. propose to leverage this additional data through the teacher-student framework where the teacher would be trained on the privileged information $\boldsymbol{x}^*$. The teacher would then be used to generate soft labels for the student which is trained on the regular input $\boldsymbol{x}$.

4.3.3 Distillation for speaker and domain robustness

Several studies have investigated the use of distillation (or generalized distillation) in the context of speaker or domain robustness. One of the first work of this kind, by Markov and Matsui (2016), was based on parallel clean and noisy speech data where the clean speech was used as privileged knowledge. They showed a 4.65% WER reduction for the student net on Aurora2 dataset. Similarly, it has been shown that several other sources of privileged information can be leveraged to improve domain robustness, such as enhanced features for noisy conditions (Watanabe, Hori, Roux, et al., 2017), broadband recordings for narrowband ones (Fukuda et al., 2017), adult data for child speech (Li, Seltzer, et al., 2017; Asami et al., 2017), or standard language for dialects (Asami et al., 2017). For speaker robustness, Yu, Markov, et al. (2016), showed that articulatory features can be used as privileged knowledge, while Joy et al. (2017) demonstrated the usefulness of fMLLR features.

While distillation focuses on the output of the network as a source of information, Kim et al. (2017) proposed the bridge-net architectures where the output of intermediate layers of the teacher are used as additional hints to guide the training of the student. Together with the use of the recurrent units, adding information about the intermediate representations improves recognition of distant speech by 5.04% relative on the AMI corpus (using clean speech as privileged knowledge) over standard distillation.

4.3.4 Distillation versus Kullback-Leibler divergence

The KLD loss, mentioned in subsection 4.2.3, is defined as follow:

$$L_{KLD}(\boldsymbol{x}_i, \boldsymbol{y}_i) = (1-\lambda) L_{CE}(\boldsymbol{y}_i, f(\boldsymbol{x}_i, \boldsymbol{\theta})) + \lambda L_{CE}(\tilde{\boldsymbol{y}}_i, f(\boldsymbol{x}_i, \boldsymbol{\theta})), \tag{4.3}$$

where $f(\boldsymbol{x}_i, \boldsymbol{\theta})$ is the function computing the output values of the network for input $\boldsymbol{x}_i$. Ignoring the regularization term in Equation 4.2, we can see that both equations are equivalent when a temperature of 1 is used. Distillation with temperature is thus an extension of KLD regularization where the temperature allows to soften the output of the teacher, hence giving more importance to the alternative classes.

4.4 Bias undoing

The following section will present a first approach I proposed to improve the speaker/domain robustness of DNN-based acoustic models. This strategy is directly inspired by the regularized MTL paradigm (Evgeniou and Pontil, 2004) and ensuing work (Khosla et al., 2012; Badino et al., 2017), where the approach has been used to build more robust speaker- or domain-independent models (even though they did so in contexts different from speech recognition). To motivate the proposed model, I will first present the relevant literature.

4.4.1 Related work

Evgeniou and Pontil (2004) proposed a new framework for MTL called regularized MTL. However, it should be noted that their definition of a task is broader than what is usually intended in ASR. Within the speech recognition community, two tasks are usually considered different if they operate on different input and/or output space. This is for example the case of the multi-task architecture used for multilingual training (Ghoshal et al., 2013; Heigold et al., 2013; Huang, Li, et al., 2013). Evgeniou and Pontil on the other hand, extend this

definition to situations where the input and output spaces are the same but the distributions used to sample the data are different (which would rather be seen as multi-condition training within the ASR community). Hence, each learning task $t \in \{1,\dots,T\}$ can be defined by a different distribution $\mathcal{D}_t$ over the space $\mathcal{X} \times \mathcal{Y}$, where $\mathcal{X}$ and $\mathcal{Y}$ are the input and output spaces respectively. Given access to a sample $D_t \sim \mathcal{D}_t$ of examples from each distribution, the goal is to learn task-specific functions $f_t : \mathcal{X} \to \mathcal{Y}$. To formalize the relation between the tasks, they define a "mean" or "canonical" function f_0 and its set of parameters $\boldsymbol{\theta}_0$. The relation between the task-specific functions f_t parameterized by $\boldsymbol{\theta}_t$ and the canonical function f_0 is defined by the distance $\boldsymbol{v}_t$ between their sets of parameters such that:

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_0 + \boldsymbol{v}_t. \tag{4.4}$$

The smaller the $\boldsymbol{v}_t$ vectors are, the closer the tasks. The explicit formulation of the relation between tasks aims at providing a regularization mechanism which should improve the performance of the task-dependent models. A parallel can easily be drawn between the tasks, as defined by Evgeniou and Pontil and the different speakers in speaker adaptation or the domains in domain adaptation. Taking this point of view, the regularized MTL paradigm is close in spirit to the SAT approach which also tries to explicitly split the acoustic model in a canonical SI part and SD sub-models that are jointly trained.

Khosla et al. (2012) made the same parallel in their work on dataset bias for computer vision. The main assumption is that each dataset (constituting a different domain) is a biased version of the full visual world. They then try to learn an unbiased function f_0 through the regularized MTL paradigm. It is to be noted that contrary to Evgeniou and Pontil, the objective here is not to improve the task-dependent models. Through the canonical model, the authors seek an unbiased function that should perform well on its own and give better performance on any new dataset. They thus incorporate the optimization of the canonical model f_0 in their loss, which was not present in the original proposal.

More related to speech recognition, Badino et al. (2017) applied a similar approach to the acoustic inversion problem, where the goal is to recover articulatory features from the acoustic domain. Also here, the use of the regularized MTL paradigm is motivated by the search of a speaker-independent model that would get rid of speakers' biases. While the two previously mentioned works were based on SVMs, Badino et al. extend the paradigm to DNN models and introduce a new similarity measure (referred to as O-MTL, in opposition to the approach based on the weights from Evgeniou and Pontil, referred to as W-MTL). In the O-MTL method, the distance between the different networks is based on the output

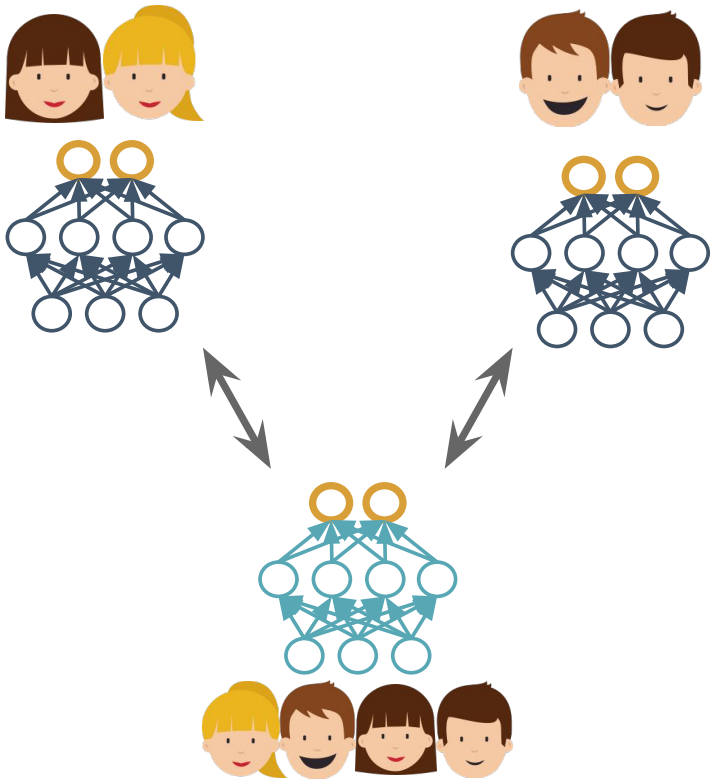

FIGURE 4.1 Illustration of the O-MTL approach applied to gender robustness. The two gender-dependent models are represented at the top while the gender-independent model is illustrated at the bottom. The arrows represent the MTL regularization loss that relates the output of the different models.

of the networks only and no parameter is shared. In this respect, the O-MTL approach is much closer to distillation or KLD-based regularization. Though, a major difference with distillation is that the SI net and the SD models are optimized in parallel. No distinction is made between teacher and student, and the two types of networks can learn from one another. Also, the way the problem is structured as a unique SI model versus several SD models each covering a speaker is new.

4.4.2 Methodology

The approach followed here takes direct inspiration from the O-MTL strategy from Badino et al. and uses it to improve the speaker/domain robustness of DNNs in acoustic modeling (illustrated in Figure 4.1).

Let $D = \{(\boldsymbol{x}_1, \boldsymbol{y}_1), \ldots, (\boldsymbol{x}_N, \boldsymbol{y}_N)\} \subset \mathcal{X} \times \mathcal{Y}$ be the set of examples composing the dataset. The reference output $\boldsymbol{y}_i$ is a one-hot vector where the entry corresponding to the reference state is set to 1 and all other entries are set to 0. Let $t \in \{1, \ldots, T\}$ represent the different tasks (speakers or domains) present in the dataset. Each example $(\boldsymbol{x}_i, \boldsymbol{y}_i)$ can be assigned to a unique task.

The main objective is to train a task-independent model

$$f_0 : \mathcal{X} \to \mathcal{Y} \tag{4.5}$$

that predicts the phoneme (or triphone or senone) category $\boldsymbol{y}_i$ of an acoustic feature vector $\boldsymbol{x}_i$. This correspond to the usual task assigned to the DNN model in acoustic modeling. The function f_0 is defined by a set of parameters $\boldsymbol{\theta}_0$.

We additionally consider task-dependent models

$$f_t : \mathcal{X} \to \mathcal{Y}, \tag{4.6}$$

where each model's parameters $\boldsymbol{\theta}_t$ are estimated from the set $D_t = \{(\boldsymbol{x}_1^t, \boldsymbol{y}_1^t), \ldots, (\boldsymbol{x}_{n_t}^t, \boldsymbol{y}_{n_t}^t)\}$ of examples belonging to task t, such that $D = \bigcup_{t=1}^{T} D_t$ and each subset D_t follows a different distribution $\mathcal{D}_t$ ($D_t \sim \mathcal{D}_t$).

One of the main difference with the acoustic inversion problem is that the DNNs trained here output a probability distribution (enforced through the softmax activation function used for the output layer), when the acoustic inversion model estimates real valued variables instead. The square loss originally considered by Badino et al. (2017) is thus replaced by the cross-entropy loss, as usually done in ASR. The overall optimization objective can now be defined as finding the set of parameters $\{\boldsymbol{\theta}_0, \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_T\}$ that minimize:

$$(1-\lambda)L(\boldsymbol{\theta}_0, \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_T) + \lambda R_{MTL}(\boldsymbol{\theta}_0, \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_T) + R_{L2}(\boldsymbol{\theta}_0, \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_T), \tag{4.7}$$

where L correspond to the usual cross-entropy loss for the task-dependent and task-independent models, R_{MTL} is the multi-task regularization term and R_{L2} corresponds to standard l_2-norm regularization of the networks parameters. As can be seen from Equation 4.7, $\lambda \in [0,1]$ allows to control the trade off between the phoneme prediction task and the MTL regularization. Hence, more regularization can be applied to the different models, lowering the constraint to match the reference labels. At one extreme, when $\lambda = 0$, no regularization is applied and the models are trained independently. On the contrary, if $\lambda = 1$, phoneme classification is completely ignored and only the distance between the output of the networks

is considered. The three components of the loss are finally defined as:

$$L(\boldsymbol{\theta}_0, \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_T) = \sum_{i=1}^{N} L_{CE}(\boldsymbol{y}_i, f_0(\boldsymbol{x}_i, \boldsymbol{\theta}_0)) + \sum_{t=1}^{T} \sum_{i=1}^{n_t} L_{CE}(\boldsymbol{y}_i^t, f_t(\boldsymbol{x}_i^t, \boldsymbol{\theta}_t)), \tag{4.8}$$

$$R_{MTL}(\boldsymbol{\theta}_0, \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_T) = \sum_{t=1}^{T} \sum_{i=1}^{n_t} \|z_t(\boldsymbol{x}_i^t, \boldsymbol{\theta}_t) - z_0(\boldsymbol{x}_i^t, \boldsymbol{\theta}_0)\|_2^2, \tag{4.9}$$

$$R_{L_2}(\boldsymbol{\theta}_0, \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_T) = \lambda_{L2_{DI}} \|\boldsymbol{\theta}_0\|_2^2 + \lambda_{L2_{DD}} \sum_{t=1}^{T} \|\boldsymbol{\theta}_t\|_2^2, \tag{4.10}$$

where $z_0(\cdot)$ and $z_t(\cdot)$ are the functions computing the logits (the output of the final layer before the softmax) of the domain-independent (DI) and domain-dependent (DD) nets respectively. I also tried using the output of the networks directly (additionally introducing the temperature parameter proposed by Hinton, Vinyals, et al.) but preliminary experiments showed better results using the logits.

4.4.3 Experimental setup

Dataset

The experiments were carried on TIMIT, as the small size of the dataset allows to iterate rapidly and explore many alternative architectures or configurations of the hyperparameters. For all experiments, we used only `SI` and `SX` sentences (see subsection 3.2.1). The amount of data per speaker being too small though, speaker adaptation was not feasible. Also, the method would have required to train one network per speaker which would have been impractical. I thus decided to use genders as the domains.

For each frame, 40 log mel-scale filter bank coefficients plus energy are extracted over windows of 25 ms, with a stride of 10 ms. First and second derivatives are also calculated giving 123 features per frame. A context of 11 frames (5 from both sides) is used as input to the networks, corresponding to 1353 values in total. The target labels correspond to 3-state monophones, that is 183 values overall. For scoring, the set of 61 phonemes is collapsed to 39 phones after decoding, as done by Lee and Hon (1989).

Baseline

Both DI and DD baseline models (later referred to as STL models, in opposition to the MTL procedure) follow the same architecture and are composed of 4 hidden layers of 2000 units, with ReLU activation function for the intermediate layers and softmax activation for the

TABLE 4.1 Hyperparameters used to train the acoustic DNN-HMM baseline on TIMIT.

Parameter	Value
Input size	1353
Output size	183
Number of hidden layers	4
Number of units per hidden layer	2000
Mini-batch size	500
Momentum	0.5
Initial learning rate	0.3
Decay rate	0.75
L2-norm regularization weight	0.0003

output one. The models are trained with mini-batch SGD with momentum and a decaying learning rate. Regularization based on l_2-norm is used and early stopping is applied after 3 epochs of decreased performance on the validation set. Table 4.1 summarizes the values of the different hyperparameters.

O-MTL approach

One of the main limitation in the use of the O-MTL approach is that each task is represented by a different network. As the number of tasks increases, training all those models in parallel can become unmanageable in terms of memory. To mitigate that, I re-used the alternate training procedure proposed by Badino et al., where $\boldsymbol{\theta}_0$ and $\boldsymbol{\theta}_t$ are trained alternatively while the other set of parameters is kept fixed. The training procedure is summarized in Algorithm 1.

Algorithm 1 O-MTL training procedure

1: Initialize $\boldsymbol{\theta}_0$ and each $\boldsymbol{\theta}_t$ with the weights of their respective STL model, saved after a number e_{init} of epochs (optimal results are obtained using the weights of a net trained only for few epochs).
2: **while** early-stopping criterion is not reached **do**
3: Train each f_t model for 1 epoch keeping $\boldsymbol{\theta}_0$ fixed.
4: Train f_0 for 1 epoch keeping all $\boldsymbol{\theta}_t$ fixed.
5: **end while**

Initializing the weights of the different networks (step 1) by first training them with the STL strategy for few epochs gave slightly better results. Similar pretraining strategy didn't improve performance on the baseline (not shown here).

TABLE 4.2 Hyperparameters used for the O-MTL approach on TIMIT dataset.

Parameter	**Value**
e_{init}	3
Initial learning rate	0.6
λ	0.2
$\lambda_{L2_{DI}}$	0.0003
$\lambda_{L2_{DD}}$	0.0003

The loss actually optimized in steps 3 and 4 of Algorithm 1 are given in Equation 4.11 and Equation 4.12. The architecture and the optimization strategy are otherwise kept identical to the baseline model. Table 4.2 summarizes the values of the hyperparameters that gave the best results for the O-MTL approach.

$$L_{DI}(\boldsymbol{\theta}_0) = (1-\lambda)\sum_{i=1}^{N} L_{CE}(\boldsymbol{y}_i, f_0(\boldsymbol{x}_i, \boldsymbol{\theta}_0)) + \lambda \sum_{t=1}^{T}\sum_{i=1}^{n_t} \|z_t(\boldsymbol{x}_i^t, \boldsymbol{\theta}_t) - z_0(\boldsymbol{x}_i^t, \boldsymbol{\theta}_0)\|_2^2 + \lambda_{L2_{DI}} \|\boldsymbol{\theta}_0\|_2^2 \tag{4.11}$$

$$L_{DD}(\boldsymbol{\theta}_t) = (1-\lambda)\sum_{i=1}^{n_t} L_{CE}(\boldsymbol{y}_i^t, f_t(\boldsymbol{x}_i^t, \boldsymbol{\theta}_t)) + \lambda \sum_{i=1}^{n_t} \|z_t(\boldsymbol{x}_i^t, \boldsymbol{\theta}_t) - z_0(\boldsymbol{x}_i^t, \boldsymbol{\theta}_0)\|_2^2 + \lambda_{L2_{DD}} \|\boldsymbol{\theta}_t\|_2^2 \tag{4.12}$$

4.4.4 Results

Table 4.3 reports the validation frame error rate (FER) of the DI and DD baselines as well as the best O-MTL, for which the DI and DD sub-nets are evaluated separately. Results of DI models are presented for all data or split by gender. For DD models, I use the models trained on each gender to compute the FER on corresponding validation data. I additionally report the weighted sum of these scores, to allow better comparison with DI models.

We can first observe that the gender-dependent STL models perform worse than their gender-independent counterpart. It is likely that TIMIT is too small to allow estimation of good gender-dependent models. The additional data provided to the gender-independent model compensate for the harder task it faces, allowing it to do $\sim 2.9\%$ relatively better than the combination of gender-dependent models.

TABLE 4.3 FER (%) of the single- and multi-task models on TIMIT validation set.

Domain dependence	**Training strategy**	**Validation FER (%)**		
		All	**F**	**M**
DI	STL	35.95	36.80	35.51
DI	O-MTL	**34.90**	**35.80**	**34.41**
DD	STL	37.04	39.35	35.82
DD	O-MTL	**36.09**	**38.22**	**34.90**

TABLE 4.4 FER (%) of the gender-dependent single- and multi-task models on TIMIT validation set. While the models are trained on data from one gender only, I report evaluation on data from both genders.

Training gender	**Training strategy**	**Validation FER (%)**		
		All	**F**	**M**
F	STL	49.51	39.35	54.87
	O-MTL	**47.20**	**38.22**	**51.95**
M	STL	41.58	52.48	35.82
	O-MTL	**40.03**	**49.60**	**34.97**

Despite that, we observe that both DI and DD models can benefit from the O-MTL approach, performing $\sim 2.9\%$ and $\sim 2.6\%$ relatively better than their STL equivalent.

In order to evaluate the impact of the O-MTL strategy on the generalization of the DD models, I also performed a more detailed analysis of their performance on the different domains. Table 4.4 reports the validation FER of gender-dependent models trained with the STL or O-MTL strategy, when evaluated on each gender (also the one they were not trained on). We can see that the improvement of those models on the gender they did not see at training time ($\sim 5.3\%$ and $\sim 5.5\%$ respectively for the models trained on female and male data) is about twice the improvement obtained on there own gender ($\sim 2.9\%$ and $\sim 2.4\%$ respectively). I hypothesize that this improvement in generalization can be attributed to the guidance of the DI model.

Finally, in Table 4.5, I report the FER and PER on the test set of the best DI models trained following the STL or O-MTL procedures. Unfortunately, the improvement obtained for the FER (about $\sim 3.2\%$) does not completely carry over to the PER, where we obtain only $\sim 1.5\%$ of relative improvement.

TABLE 4.5 FER and PER (%) of the gender-independent single and multi-task models on TIMIT test set.

Training strategy	Test FER (%)	Test PER (%)
STL	36.98	22.05
O-MTL	**35.81**	**21.71**

4.4.5 Conclusion

The O-MTL training approach to domain robustness showed mitigated results here. I managed to obtain $\sim 3.2\%$ of relative improvement on the test FER but this only partially carries over to the PER ($\sim 1.5\%$ of relative improvement). It may be though that TIMIT is too small for proper estimation of gender-dependent models, as shown by the poorer performance they obtain compared to a DI model. It would be interesting then to see how this approach performs on bigger datasets and possibly with different sources of variation (especially noise).

Alternatively, it could be interesting to try a similar approach in the context of linear networks (LIN, LHN or LON), where task-independent and task-dependent linear transformations could be considered, instead of full networks. This would probably be better suited to a small dataset such as TIMIT.

Finally, while I consider here the use of the canonical DI model, improvement is also observed for the DD models. Instead of aiming for a stronger task-independent model, one could rather try to improve performance for the task-dependent model corresponding to the target application. Applying this approach to domain adaptation in the scope of iCubrec pipeline, one option would then be to train a separate model for source and target domains, where the target domain is covered by VoCub dataset. While VoCub is small and overfitting is observed when training a model on this dataset only, the O-MTL approach developed here could offer a way to regularize it and improve its performance.

4.5 Task-aware training

One of the limitations of the O-MTL approach I just presented is the necessity to train a separate model for each task. Also, the generalization of the model to new tasks that are relatively different from the ones seen during training may be poor (an example would be a model trained on several noise conditions that faces a completely new one at runtime).

The speaker- or domain-aware training approach proposes to circumvent these issues by providing a representation of the speaker or domain to the network in order to help it factor out this source of variability more easily. One can also hope that, when faced with a new condition, the model can leverage on this information to better generalize to it.

4.5.1 Methodology

Inspired by this line of research, I propose a second approach where the task dependent models are replaced with a single task-aware model. Making a parallel with distillation, I will refer to this model as the teacher, as it has access to privileged knowledge through the task embedding that is concatenated to its input. The other network, which does not have access to the task embedding, is thus referred to as the student network. It is to be noted though that unlike the distillation paradigm, teacher and student both learn from one another. Equation 4.7, defining the O-MTL loss function, can be rewritten in this case as:

$$(1-\lambda)L(\boldsymbol{\theta}_T,\boldsymbol{\theta}_S)+\lambda R_{MTL}(\boldsymbol{\theta}_T,\boldsymbol{\theta}_S)+\lambda_{L2}R_{L2}(\boldsymbol{\theta}_T,\boldsymbol{\theta}_S), \tag{4.13}$$

where $\boldsymbol{\theta}_T$ and $\boldsymbol{\theta}_S$ are the parameters of the teacher (f_T) and student (f_S) models respectively, and λ_{L2} is a unique weight applied to R_{L2} (instead of the individual weights $\lambda_{L2_{DI}}$ and $\lambda_{L2_{DD}}$ used previously).

I discovered through the first set of experiments on bias undoing that optimizing hyper-parameters for such a criterion is quite hard. Indeed, as λ is modified, the scale of L and R_{MTL} terms changes and the scale of the full loss with them. This is exemplified by the much higher initial learning rate used with the O-MTL approach (0.6) compared to the STL one (0.3). An alternative and equivalent loss (apart for the values of the hyperparameters that should be scaled accordingly) is thus considered here. The scale of the main term, L, is kept constant, with the hope that the learning rate will then be less dependent on λ, giving:

$$L(\boldsymbol{\theta}_T,\boldsymbol{\theta}_S)+\lambda R_{MTL}(\boldsymbol{\theta}_T,\boldsymbol{\theta}_S)+\lambda_{L2}R_{L2}(\boldsymbol{\theta}_T,\boldsymbol{\theta}_S). \tag{4.14}$$

The different terms are in turn defined as:

$$L(\boldsymbol{\theta}_T, \boldsymbol{\theta}_S) = \sum_{i=1}^{N} L_{CE}(\boldsymbol{y}_i, f_T(\boldsymbol{x}_i^*, \boldsymbol{\theta}_T)) + L_{CE}(\boldsymbol{y}_i, f_S(\boldsymbol{x}_i, \boldsymbol{\theta}_S)), \tag{4.15}$$

$$R_{MTL}(\boldsymbol{\theta}_T, \boldsymbol{\theta}_S) = \sum_{i=1}^{N} \|z_T(\boldsymbol{x}_i^*, \boldsymbol{\theta}_T) - z_S(\boldsymbol{x}_i, \boldsymbol{\theta}_S)\|_2^2, \tag{4.16}$$

$$R_{L_2}(\boldsymbol{\theta}_T, \boldsymbol{\theta}_S) = \|\boldsymbol{\theta}_T\|_2^2 + \|\boldsymbol{\theta}_S\|_2^2, \tag{4.17}$$

where $z_T(\cdot)$ and $z_S(\cdot)$ correspond to the functions computing the logits of the teacher and student networks respectively. Again here, I also tried using the output of the networks directly (introducing again the temperature parameter) but preliminary experiments showed better results using the logits.

The student model is the one we intend to use at runtime. Even though it is expected to perform worse than the teacher model, it has some advantages. To reliably estimate the embeddings needed by the teacher, enough data should be collected and a first pass on the data should be performed to extract the relevant information. This adds latency and complexity to the pipeline, which is a drawback for online embedded ASR systems.

4.5.2 Experimental setup

Again, I use TIMIT dataset for the experiments. As the new method do not resort to separate task-dependent models though, it is possible to work directly on speaker robustness instead of grouping them by gender. As for the task representation, I opted for i-vectors. They have been shown to be a very effective representation of speakers but also acoustic environment and channel. They can thus address most of the sources of variability we are concerned with.

I-vectors extraction

To extract the i-vectors, the MSR Identity Toolbox[1] has been used, which is based on MATLAB. After estimation of the GMM-based universal background model (UBM), the total variability subspace is computed and LDA is applied to project the i-vectors onto a lower dimensional space with good speaker class-separability.

In addition to i-vectors computed at the utterance-level, I also experimented with speaker-level ones. To obtain them, I simply take the mean of the utterance-level i-vectors of each speaker.

[1] https://www.microsoft.com/en-us/download/details.aspx?id=52279

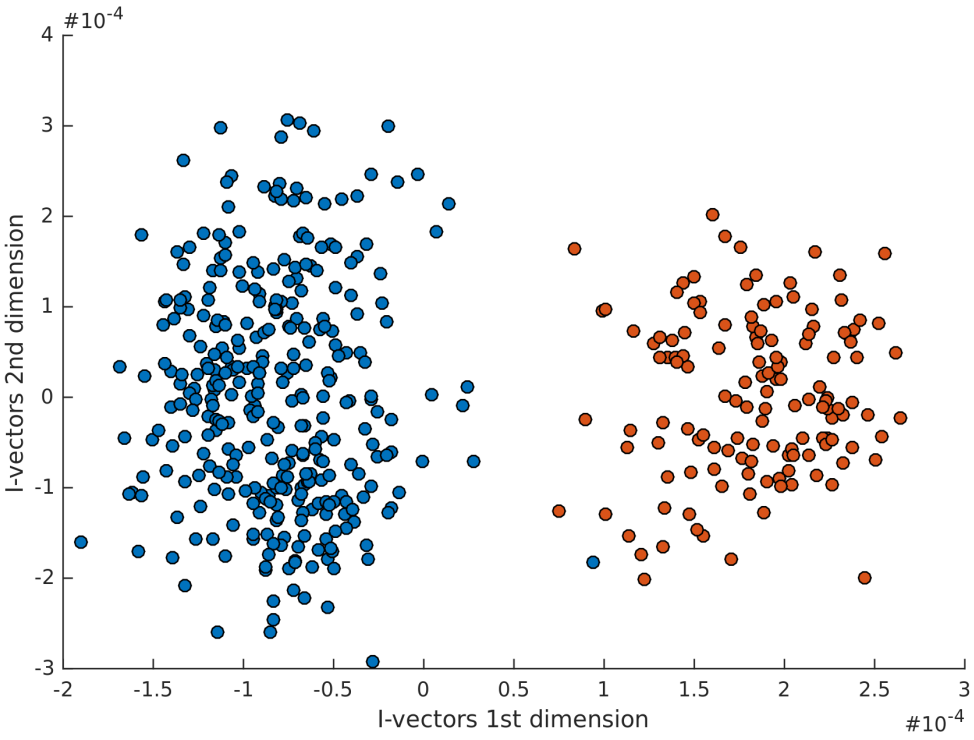

FIGURE 4.2 The 1^{st} and 2^{nd} dimensions of the speaker i-vectors after PLDA. Female speakers are represented in orange and male speakers in blue.

As can be seen from Figure 4.2, the first dimension found by the dimensionality reduction algorithm allows to discriminate speakers by gender nearly perfectly. Apart from one male speaker, the two genders would be linearly separable just based on this dimension. This confirms quantitatively gender as one of the main sources of variability in our data and thus its use in the previous set of experiments.

Baselines

Two baselines are considered here. The first one is the same as the STL model presented in previous section and will be the reference for the student network. The second one, trained using the speaker-aware training (SAwT) procedure, has i-vectors appended to its input and will serve as a baseline for the teacher model. I will refer to them as the `STL` and `SAwT` models respectively. Hyperparameters are the same as the ones presented in Table 4.2, except for the input size which is increased by the size of the i-vectors in the case of the SAwT model.

Training procedure

An additional difference with the precedent set of experiments is the introduction of a new training procedure. Instead of the parallel procedure presented precedently, where I alternatively train each network for one epoch, I use here a fully alternated training procedure. In this case, I train one network completely (until early stopping criteria is met) before extracting the logits needed to train the other network. This second network is in turn trained completely, before logits are extracted to retrain the first model. The motivation for this new procedure is that the performance of each network is best at the end of the training and so should be the information provided by the logits. Providing these final logits to the other model from the beginning may thus help it reach a better minimum. Also, using the same secondary targets throughout the training of the network may lead to a better optimum. When the two networks are trained in parallel, the information they provide to each other drifts as the training progresses, which may compromise the training procedure.

The loss optimized by teacher and student networks with this new training procedure are given in Equation 4.18 and Equation 4.19. The architecture and the optimization strategy are otherwise kept identical to the baseline models.

$$L_S(\boldsymbol{\theta}_S) = \sum_{i=1}^{N} L_{CE}(\boldsymbol{y}_i, f_S(\boldsymbol{x}_i, \boldsymbol{\theta}_S)) + \lambda \sum_{i=1}^{N} \|z_S(\boldsymbol{x}_i, \boldsymbol{\theta}_S) - z_T(\boldsymbol{x}_i^*, \boldsymbol{\theta}_T)\|_2^2 + \lambda_{L2}\|\boldsymbol{\theta}_S\|_2^2 \tag{4.18}$$

$$L_T(\boldsymbol{\theta}_T) = \sum_{i=1}^{N} L_{CE}(\boldsymbol{y}_i, f_T(\boldsymbol{x}_i^*, \boldsymbol{\theta}_T)) + \lambda \sum_{i=1}^{N} \|z_S(\boldsymbol{x}_i, \boldsymbol{\theta}_S) - z_T(\boldsymbol{x}_i^*, \boldsymbol{\theta}_T)\|_2^2 + \lambda_{L2}\|\boldsymbol{\theta}_T\|_2^2 \tag{4.19}$$

The new training procedure (illustrated in Figure 4.3) can be summarized as follow. I start from the `SAwT` baseline, which is used as the first teacher T_0. It is the only model used in the O-MTL procedure that does not use the loss described in Equation 4.13. T_0 is used to extract the logits that will be used as secondary targets for the first student model S_1. Once trained, the S_1 model is in turn used to obtain logits that will serve as secondary targets for the second teacher T_1 (as we consider the `SAwT` as the first teacher). And we continue alternating the training of student and teacher networks likewise, for a fixed number of iterations or until

they stop improving from one step to the next. This training procedure is summarized in Algorithm 2.

Algorithm 2 Task-aware training procedure

1: Initialize the teacher model (T_0) with the parameters of the `SAwT` baseline.
2: **while** stopping criterion is not reached **do**
3: (Re)initialize the parameters $\boldsymbol{\theta}_S$ of the student model.
4: Train the student model keeping the teacher fixed, optimizing loss L_S.
5: Reinitialize the parameters $\boldsymbol{\theta}_T$ of the teacher model.
6: Train the teacher model keeping the student fixed, optimizing loss L_T.
7: **end while**

The new training procedure makes the relation with distillation even more visible. The remaining differences are the presence of secondary targets for the teacher and the alternate training procedure that refines the different models over several iterations.

4.5.3 Results

Figure 4.4 shows the evolution of the validation FER for the `SAwT` model as a function of: (1) the size of the i-vector, (2) the use of speaker or utterance level i-vectors and (3) the use of LDA or not. Using LDA clearly improves the effect of the i-vectors. The level at which i-vectors are computed has less impact on the performance but speaker-level i-vectors consistently outperform utterance-level ones, with and without LDA. The best results are obtained for speaker-level i-vectors of size 50 with LDA. As shown in Table 4.7, the best `SAwT` model improves over the validation FER of the `STL` baseline by 3.8% relative.

Figure 4.5 and Figure 4.6 show the evolution of the validation performance (FER) of the student and teacher networks respectively, over 10 iterations and for $\lambda \in \{0.1, 0.3, 0.5, 0.7\}$. Higher values of λ have been tried but make the training diverge. Sometimes, experiments also diverged for lower values of λ, after a few iterations. In the experiments presented here, this happened for $\lambda = 0.3$ and 0.5, as can be seen by the missing points in the figures. This could probably be avoided with a better choice of hyperparameters, or alternatively, with a different optimization algorithm. Adaptive algorithms, such as Adam presented in subsection 2.5.3 could be a good choice here.

We can first notice that the evolution of the performance do not always follow a stable trajectory over iterations and tend to fluctuate, especially for the teacher network. A good example is the evolution of the teacher network with $\lambda = 0.3$, where the FER seems to increase until the 7^{th} iteration and then falls down for two iterations reaching its minimum at the 9^{th} iteration.

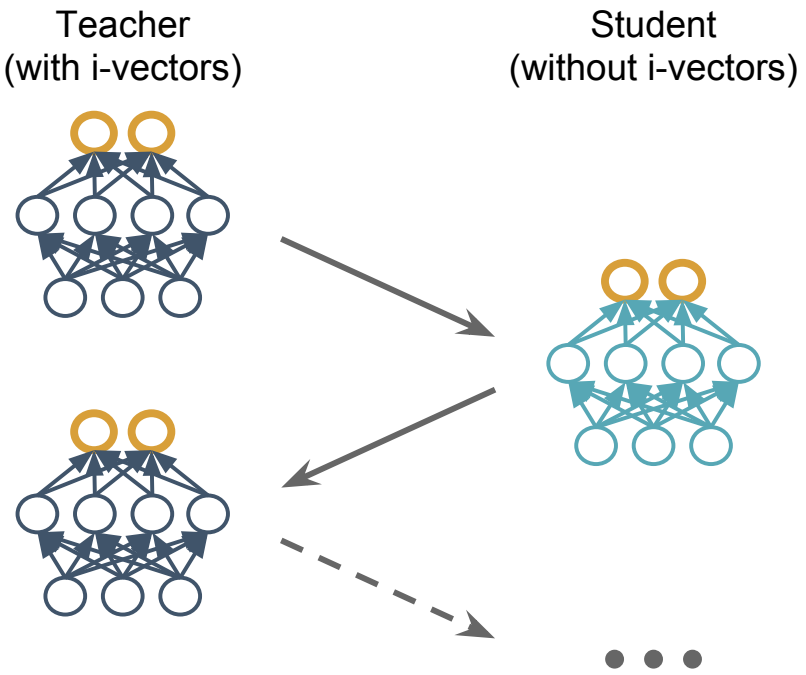

FIGURE 4.3 Illustration of the fully alternated training procedure. The first teacher is initialized with the SAwT baseline, after which student and teacher are trained alternatively using the logits of the previously trained model, until convergence is reached.

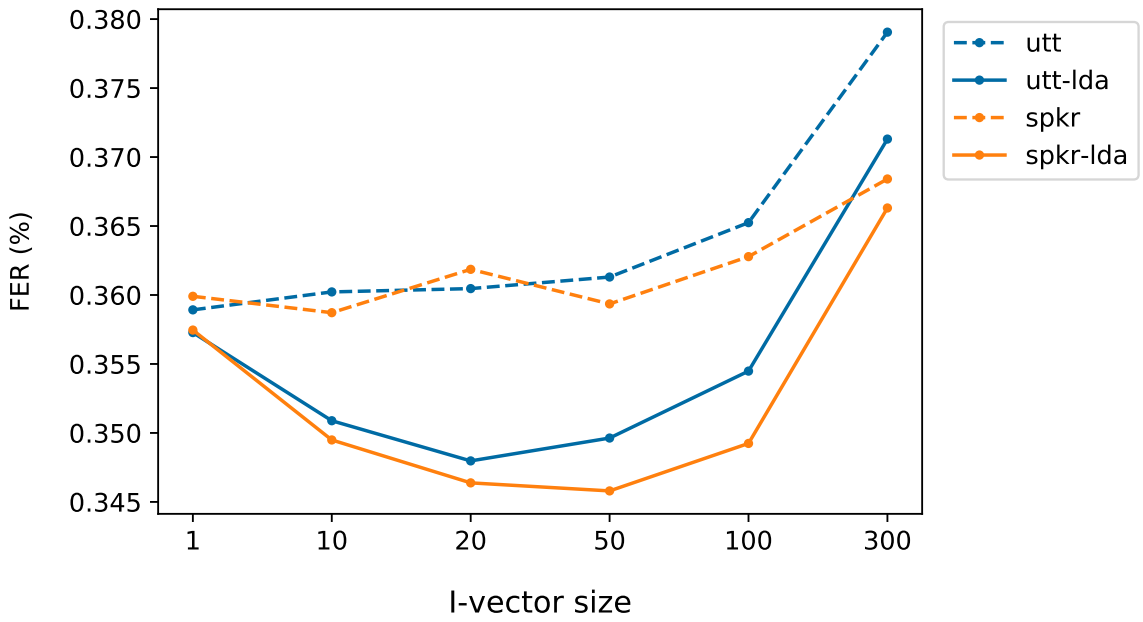

FIGURE 4.4 FER (%) of the SAwT baseline on TIMIT validation set, as a function of the i-vector's dimension. Results are presented for utterance-level (`utt`) and speaker-level (`spkr`), with or without lda.

Globally, increasing λ tends to give better results for the student network. As λ increases, more and more iterations are needed to reach the optimum FER. On the contrary, the teacher network gives better results with lower values of the hyperparameter. This suggests that teacher and student may need different values of λ to achieve best performance.

Table 4.6 summarizes the validation performance of the teacher and student networks for the best iteration. The student network reaches 35.52% of FER for $\lambda = 0.7$ while the teacher network reaches 34.39% of FER for $\lambda = 0.1$. Table 4.7 further compares the best student and teacher models with the 2 baselines. We can see that the best student model improves on the STL baseline by 1.2% relative, while the teacher improves on the SAwT baseline by 0.5%. Hence, despite the O-MTL being effective in improving both the teacher and student models, the improvement remains quite modest.

Finally, the PER of the STL baseline system and the best student model on the test set is given in Table 4.8. The FER improvement obtained with the O-MTL approach on the validation set translates poorly to the PER on the test set with only 0.3% of improvement.

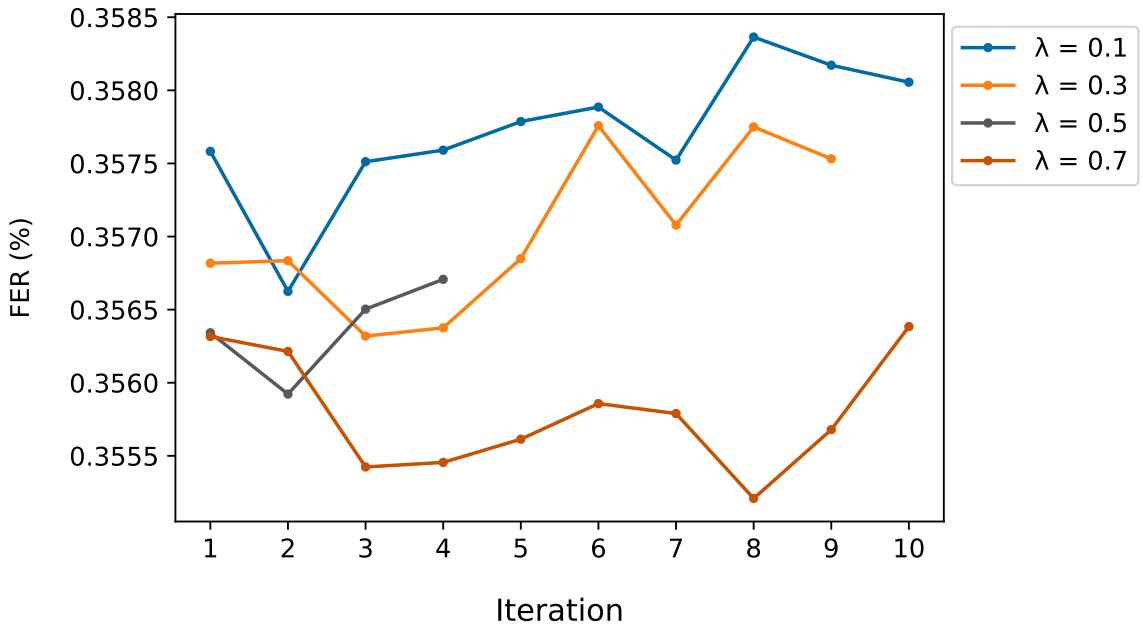

FIGURE 4.5 FER (%) of the student network on TIMIT validation set, as a function of the iteration and for different values of λ.

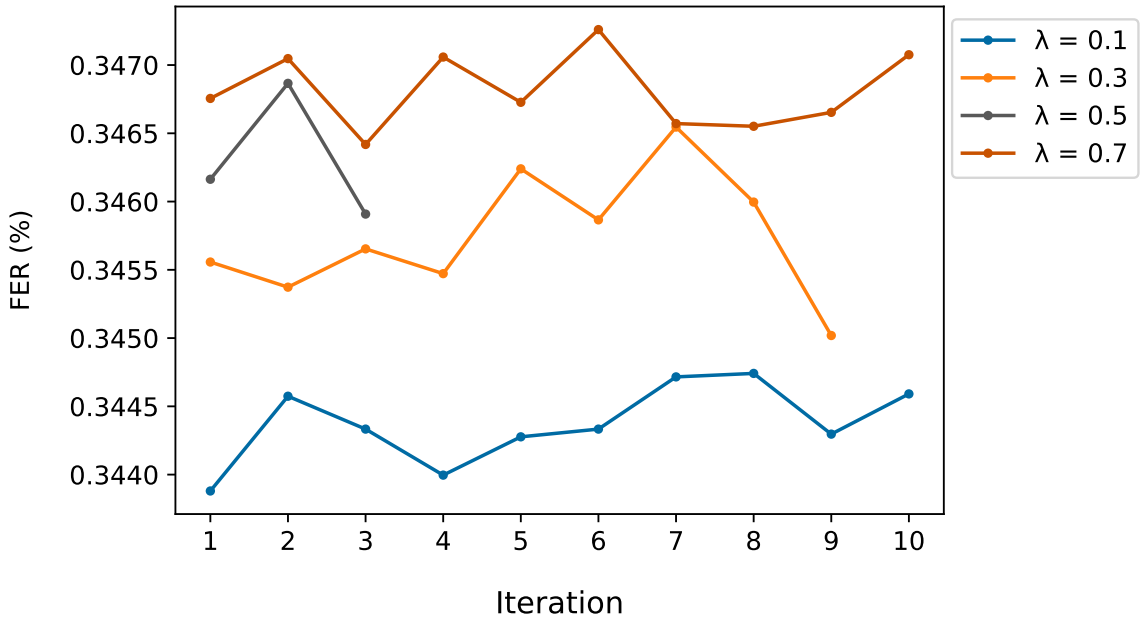

FIGURE 4.6 FER (%) of the teacher network on TIMIT validation set, as a function of the iteration and for different values of λ.

TABLE 4.6 FER (%) of the student and teacher network on TIMIT validation set, for different values of λ (each time at the best iteration).

λ	**Student**	**Teacher**
0.1	35.66	34.39
0.3	35.63	34.50
0.5	35.67	34.59
0.7	35.52	34.66

TABLE 4.7 FER (%) on TIMIT validation set using the baseline systems or the best teacher and student.

Strategy	**FER (%)**
STL	35.95
SAwT	34.58
O-MTL (student $\lambda = 0.7$)	35.52
O-MTL (teacher $\lambda = 0.1$)	34.39

TABLE 4.8 PER (%) on TIMIT coreTest set using the STL baseline system or the best student.

Strategy	**PER (%)**
STL	22.05
O-MTL (student $\lambda = 0.7$)	21.97

4.5.4 Conclusion

I this section, I proposed a different O-MTL paradigm where a speaker-aware network is trained jointly with a model that does not have access to such information. Also, instead of training the models jointly, I experimented with a new strategy where the two networks are trained fully in alternation. The success of the method is quite limited, with an improvement of only 1.2% on the validation FER for the model trained without i-vectors.

Two main reasons can explain this poor performance. First, as seen from the results obtained with the `SAwT` baseline, i-vectors yield limited improvement on TIMIT. Using speaker-aware training only improves on the baseline by 3.8%. It would then be interesting to reproduce the same kind of experiments on a different dataset. Also, the results suggest that a better optimization strategy, may lead to further improvement. The use of an asymmetric λ parameter for the teacher and the student is a first possibility. Using an adaptive optimization algorithm such as Adam is another one.

Also, as already noticed in the previous set of experiments with O-MTL on TIMIT, the performance only partially transfers to the PER. This illustrates the limit of using the FER as a proxy for the real measure of the performance when training a neural network for acoustic modeling. The use of sequence discriminative training or sequence-to-sequence models should help in that respect.

Finally, both sets of experiments where carried on the TIMIT corpus (a small dataset) and focused on gender or speaker variability. It would be interesting to test the same strategies with bigger datasets and sources of variability more related to the command recognition task I aim to address (e.g. noise, reverberation or channel). This investigation is postponed to future work.

Chapter 5

Few-shot learning with attention-based sequence-to-sequence models

The complexity of modern speech recognition systems based on DNN-HMM technology and the difficulty of gathering the necessary data can make it hard for single individuals or small companies to develop their own systems. E2E approaches have recently become popular as a means of simplifying the training and deployment of speech recognition systems, thus addressing the first issue. However, they often require large amounts of data to perform well on large vocabulary tasks. With the aim of making E2E approaches usable by a broader range of users, this chapter explores the potential use of E2E methods in small vocabulary contexts where smaller datasets may be used. A significant drawback of small-vocabulary systems is the difficulty of expanding the vocabulary beyond the original training samples – therefore I also study strategies to extend the vocabulary with only few examples per new class (few-shot learning).

The work presented in this chapter was developed during a visiting period at the Centre for Speech Technology Research (CSTR), University of Edinburgh, under the supervision of Dr. Peter Bell. It is an extended version of the work presented in Higy and Bell (2018).

After motivating the present work in section 5.1, relevant literature will be presented in section 5.2. Methodology and experiments are discussed in sections 5.4 and 5.5 respectively, and I conclude in section 5.6.

5.1 Motivations

The motivation behind this work is similar to that behind the Google Tensorflow team's release of the SC dataset (Warden, 2018) and the organization of an accompanying challenge[1]: make speech recognition technology – and more precisely command recognition – accessible to a wider audience. Together with the SC dataset, Google released a baseline classification system[2] which was used as a starting point by many challenge participants. To enable a simple classification system to be used directly without the use of time-warping or other dynamic programming algorithms, every input file in the dataset is constrained to a fixed length, something that would not be required by the more flexible standard HMM-based approaches to speech recognition. Although the fixed-length constraint is not unreasonable for a small vocabulary keyword recognition task, I was motivated to consider a more recent E2E approach – specifically the attention-based encoder-decoder architecture – as a means of allowing input of arbitrary length, whilst retaining the simplicity of a single DNN-based discriminative classifier. This approach also allows me to readily switch between sub-word (phoneme or grapheme) and word modeling by just changing the target output. The main drawback of a word-based output is the impossibility to introduce any new keyword without modifying the architecture of the network. Phoneme- and grapheme-based outputs allow to adapt more easily an existing network to new keywords as the constituent units are shared across words.

While those advantages are not fundamental in the scope of the keyword recognition task proposed with the SC dataset, the problem tackled in this thesis is broader. The command recognition task I consider is not restricted to keywords and employs longer commands, where the flexibility of the attention-based encoder-decoder architecture may prove useful. I chose to use the SC dataset, which is publicly available, as it allows the replication of my results. The dataset also has the advantage of providing a large number of examples per keyword, allowing to test the proposed approach in good conditions.

This chapter presents experiments on the use of a S2S model for a modified version of the SC task, comparing it with a more traditional CNN-HMM approach. In the literature, S2S models are usually applied on large vocabulary tasks with large datasets and it is not obvious that they will work well in my setup. An exception to this comes from work on KWS where a small set of keywords is usually considered (see e.g. He, Prabhavalkar, et al., 2017; Shan et al., 2018). However, the approaches are often specific to isolated keywords and the number of examples available per keyword is usually very large.

[1] https://www.kaggle.com/c/tensorflow-speech-recognition-challenge
[2] https://www.tensorflow.org/tutorials/sequences/audio_recognition

As mentioned previously, an obvious limitation of the small vocabulary approach I take is its flexibility (in the sense that only a closed set of commands can be recognized). This is even more the case with S2S models where the LM is an integral part of the model and can not easily be modified (as can be done with a DNN-HMM system, see e.g. the extended grammar for VoCub dataset in subsection 3.2.5). The S2S system is then confined to the list of commands defined in the original data. To alleviate this constraint, strategies are also explored to extend the set of commands with very few examples, thus addressing the few-shot learning problem (Ravi and Larochelle, 2017; Snell et al., 2017; Yang et al., 2018)).

5.2 Related work

While many papers have explored the use of E2E architectures for large vocabulary tasks, few have considered small vocabulary or low resource contexts.

Most of the work considering a small vocabulary comes from literature on KWS. In this context, commands are usually restricted to isolated keywords, thus allowing the use of approaches that would not extend easily to longer word sequences. A good example is the CNN classifier (or its feed-forward predecessor presented by Chen et al. (2014)), proposed by Google in the scope of the Kaggle challenge. When isolated short words are considered, it is indeed possible to consider simple classifiers that predict the probability of each frame to be part of one of the keywords or not. The context available to the network is then enough to make a decision. While a simple posterior handling algorithm has also been proposed to smooth the output and handle short key-phrases (Chen et al., 2014; Prabhavalkar, Alvarez, et al., 2015), its application to more complex syntactic structures is very limited (see section 2.2 for a deeper discussion). The work closer to mine is probably the one from Shan et al. (2018) which also uses an attention-based architecture. However, as for previous approaches, the proposed architecture is still a classifier. No decoder is present and the output of the attention function is directly converted into a confidence score. A second limit of this line of research is the amount of data used to train the models. Very good performance is obtained thanks to large datasets. In the work of Shan et al. (2018) for example, $\sim$1700 hours of data are used to detect a single wake-up word.

Conversely, work exists on the application of the E2E paradigm to low resource conditions, but usually considers large vocabulary tasks. A good example is the work of Rosenberg et al. (2017), which applied an attention-based encoder-decoder architecture to both ASR and KWS. They achieved competitive results on several languages for the large vocabulary

ASR task, even though they failed to surpass a DNN-HMM baseline. For KWS, a LVCSR method based on lattice-search is used which showed limited results.

Between the different E2E approaches, the attention-based encoder-decoder architecture has been shown to give better results on a LVCSR task (Prabhavalkar, Rao, et al., 2017) and will thus be used here. More precisely, the hybrid CTC/Attention model from Kim et al. (2017) will be used, which gave promising results and for which the code was readily available[3]. I will now present this architecture.

5.3 Joint CTC/Attention architecture

The hybrid CTC/Attention architecture proposed by Kim et al. (2017) and illustrated in Figure 5.1 uses a multi-task training approach where the encoder is shared by two decoders, one based on CTC and one using an attention mechanism. Both architectures are trained jointly. The motivation is to use CTC, which enforces monotonic alignments, to guide the training of the shared encoder. This helps the more powerful but also too flexible attention mechanism (see subsection 2.6.2) converge faster and to a better optimum.

More formally, the problem can be stated as trying to estimate the probability of an output sequence $\boldsymbol{y} = (y_1, \dots, y_U)$ given the input sequence $\boldsymbol{X} = (\boldsymbol{x}_1, \dots, \boldsymbol{x}_T)$. The model further conditions each individual output label on previous ones, so that the sequence probability can be factorized as:

$$P(\boldsymbol{y}|\boldsymbol{X}) = \prod_u P(y_u|\boldsymbol{X}, \boldsymbol{y}_{1:u-1}), \tag{5.1}$$

where $\boldsymbol{y}_{1:u-1} = (y_1, \dots, y_{u-1})$.

The architecture is first composed of an encoder which is shared between the CTC and the attention-based encoder-decoder network. The role of the encoder is to convert the input features $\boldsymbol{X}$ into a higher-level representation $\boldsymbol{H} = (\boldsymbol{h}_1, \dots, \boldsymbol{h}_K)$.

$$\boldsymbol{H} = Encoder(\boldsymbol{X}) \tag{5.2}$$

The encoder is also used to downsample the input so that the input length is usually smaller than the length of the hidden embedding ($K < T$).

The CTC loss has been described in section 2.5.2. I will now describe the attention and decoder components. The main difficulty of the sequence-to-sequence problem defined above is the dependence of the variable-length sequence $\boldsymbol{y}$ on another variable-length sequence,

[3] https://github.com/espnet/espnet

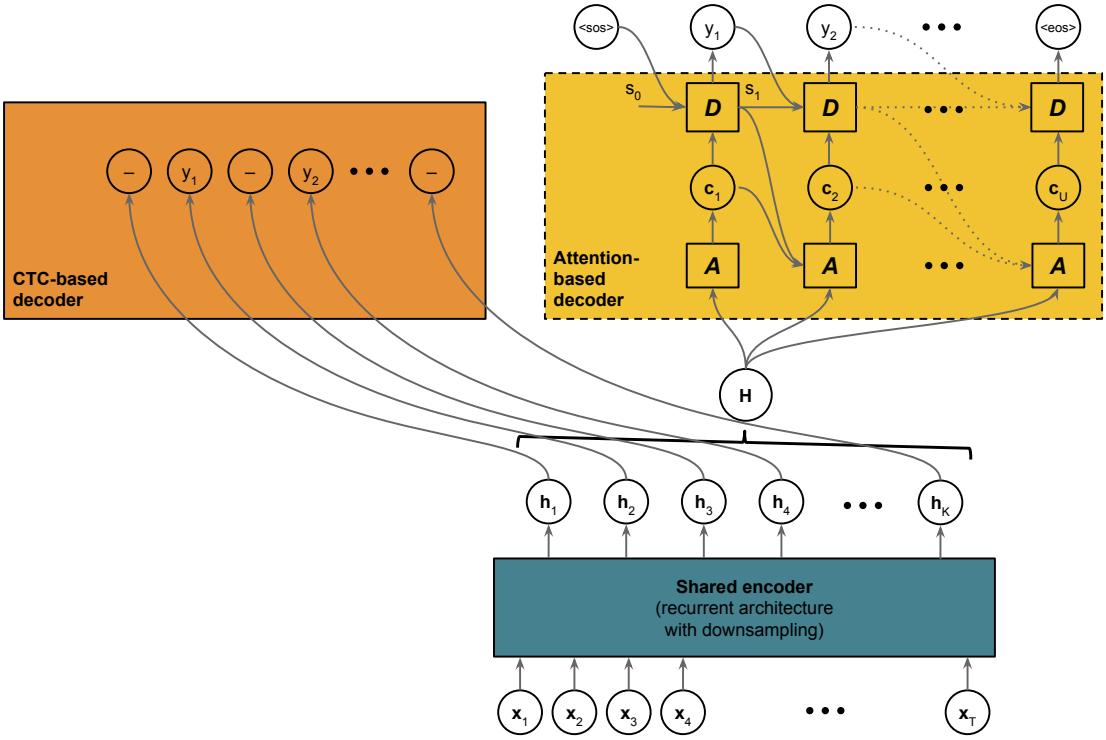

FIGURE 5.1 Architecture of the hybrid CTC/Attention model. Blocks denoted with *A* and *D* respectively represent the attention and decoder functions of the attention-based decoder.

$\boldsymbol{X}$. In the attention-based encoder-decoder architecture, this is solved by the attention mechanism, which converts the variable-length intermediate representation $\boldsymbol{H}$ into a fixed-length representation $\boldsymbol{c}_u$, based on $\boldsymbol{H}$ and the previous decoder state $\boldsymbol{s}_{u-1}$.

$$\boldsymbol{c}_u = Attention(\boldsymbol{H}, \boldsymbol{s}_{u-1}). \tag{5.3}$$

Finally, the decoder uses the output of the attention $\boldsymbol{c}_u$ as well as previous state $\boldsymbol{s}_{u-1}$ and previous predicted label y_{u-1} to predict next label y_u and update its internal state $\boldsymbol{s}_u$.

$$(y_u, \boldsymbol{s}_u) = Decode(\boldsymbol{c}_u, \boldsymbol{s}_{u-1}, y_{u-1}). \tag{5.4}$$

We will now see the three components – encoder, attention and decoder – in more details.

5.3.1 Encoder

A popular choice for the encoder is to use pyramidal BiLSTM layers (Chan, Jaitly, et al., 2016), which provide downsampling in a layer by concatenating the output of every 2 units from the layer below. Alternatively, convolutional layers can be used in combination with standard BiLSTM layers to apply downsampling. CNN-BiLSTM encoders have been shown to give better results than layers based only on LSTM or gated recurrent unit (GRU) units in E2E KWS (Shan et al., 2018). More interestingly, it has also been shown to be effective in the scope of the hybrid approach considered here (Hori et al., 2017). Both pyramidal and CNN-based BiLSTM encoders are readily available in ESPnet framework and will be compared here.

5.3.2 Attention

To combine hidden vectors $\boldsymbol{h}_k$ into a fixed length representation $\boldsymbol{c}_u$, a weighted sum is calculated for each output label as:

$$\boldsymbol{c}_u = \sum_k a_{u,k} \boldsymbol{h}_k, \tag{5.5}$$

where $\boldsymbol{a}_{u,k}$ is the attention weight applied to vector $\boldsymbol{h}_k$ at output step u. The attention vector $\boldsymbol{a}_u = (\boldsymbol{a}_{u,1}, \ldots, \boldsymbol{a}_{u,K})$ is estimated at each step based on the intermediate representation $\boldsymbol{H}$ and previous decoder state $\boldsymbol{s}_{u-1}$. Additionally, it can be conditioned on previous attention vector $\boldsymbol{a}_{u-1}$, as is the case with the location-aware attention (Chorowski, Bahdanau, Serdyuk,

et al., 2015) employed here. The attention weight $a_{u,k}$ can then be computed as:

$$\boldsymbol{F}_u = \boldsymbol{K} * \boldsymbol{a}_{u-1}, \tag{5.6}$$

$$e_{u,k} = \boldsymbol{w}^T \tanh(\boldsymbol{W}\boldsymbol{s}_{u-1} + \boldsymbol{V}\boldsymbol{h}_k + \boldsymbol{U}\boldsymbol{f}_{u,k} + \boldsymbol{b}), \tag{5.7}$$

$$a_{u,k} = \frac{\exp(\gamma e_{u,k})}{\sum_j \exp(\gamma e_{u,j})}, \tag{5.8}$$

where $*$ denotes the convolution, $\boldsymbol{f}_{u,k}$ is the convolutional feature vector, $e_{u,k}$ is the energy value, and γ is an inverse temperature or sharpening factor. The trainable parameters are: (i) the convolution kernel $\boldsymbol{K}$, (ii) the weight matrices $\boldsymbol{W}$, $\boldsymbol{V}$, $\boldsymbol{U}$ respectively applied to the previous decoder state ($\boldsymbol{s}_{u-1}$), the k^{th} output of the encoder ($\boldsymbol{h}_k$) and the k^{th} convolutional feature vector ($\boldsymbol{f}_{u,k}$), (iii) the bias $\boldsymbol{b}$ and (iv) the weight vector $\boldsymbol{w}$ that is applied to the output of the tanh function to obtain the energy. The attention weights are computed as the softmax of the energies $e_{u,k}$, which are themselves computed based on the convolutional features, the hidden features and previous decoder state.

5.3.3 Decoder

The decoder uses the fixed size representation $\boldsymbol{c}_u$ to update its internal state $\boldsymbol{s}_u$ and predict next output y_u. It is composed of a unidirectional LSTM layer followed by a linear layer and a softmax function to convert the output into a probability distribution. This can be formalized as:

$$(\boldsymbol{q}_u, \boldsymbol{s}_u) = LSTM(\boldsymbol{c}_u, \boldsymbol{s}_{u-1}, \boldsymbol{y}_{u-1}), \tag{5.9}$$

$$y_u \sim Softmax(Lin(\boldsymbol{q}_u)), \tag{5.10}$$

where q_u is the output of the LSTM layer.

Two additional tokens are added to the output labels. The start-of-sequence token (`<sos>`) is used to initialize the sequence while the end-of-sequence token (`<eos>`) indicates that sequence decoding completed.

The attention-based decoder loss uses the cross-entropy criterion based on the ground truth output sequence $\hat{y}$. This loss can be defined as:

$$L_{att} = -\ln P(\hat{\boldsymbol{y}}|\boldsymbol{X}) = -\sum_u \ln P(\hat{y}_u|\boldsymbol{X}, \hat{\boldsymbol{y}}_{1:u-1}) \tag{5.11}$$

5.3.4 Combining the two approaches

The overall loss L_{MTL} proposed by Kim et al. to train the full model is finally defined as the combination of the CTC loss L_{CTC} and the attention loss L_{att} using a weighting parameter λ:

$$L_{MTL} = \lambda L_{CTC} + (1-\lambda) L_{att}. \tag{5.12}$$

Hori et al. (2017) also proposed a way to combine the prediction of the CTC and attention-based architectures at test time in order to further improve the performance of the hybrid model.

5.4 Methodology

The main task considered here corresponds to the one proposed in Tensorflow's SC challenge presented with the dataset in subsection 3.2.4, and which evaluates keyword classification. The dataset is composed of more than 105K examples of 35 possible keywords, with each example being 1 second long. Among the 35 keywords represented, 10 are used as target commands for detection (see `org_kwd` set in Table 5.2 below for the complete list) while the other are used as OOV words to test for false detections. In addition to the keywords, two additional classes are considered: (i) a `_silence_` category corresponding to records free of speech, and (ii) an `_unknown_` category corresponding to records containing speech that is none of the keywords and populated with the remaining 25 keywords.

5.4.1 TensorFlow example code

Together with the Kaggle challenge mentioned earlier, TensorFlow team released example code to serve as a starting point for the challenge participants. This code is also the one used by Warden (2018) to report baseline results.

The model is an all neural classification system that operates on the whole example at once (leveraging on the fixed length of the records) and directly outputs a prediction score for each of the 12 classes. The input is composed of 40 MFCCs computed over windows of 30ms, with a stride of 10ms by default. Inspired from the architecture proposed by Sainath and Parada (2015), the network is composed of two 2D (across both time and frequency) convolutional layers of 64 filters over patches of 8x20 and 4x10 respectively. The first convolution is followed by a max pooling layer of 2x2 with a stride of (2, 2). Dropout is also

TABLE 5.1 Classification error (%) of the baseline systems on the 12 categories for version 1 of the SC dataset.

Model	**Classification error (%)**	
	Validation set	**Test set**
Tensorflow classifier	10.8	10.9
CNN-HMM	2.82	3.63

applied after each convolutional layer. A final linear layer followed by the softmax function gives the final prediction.

5.4.2 CNN-HMM baseline

As an alternative to the classification system just mentioned, a baseline based on a more classical CNN-HMM ASR system is described now. This system does not have the simplicity of Tensorflow's code but is much more flexible. The experiments presented here are based on code that was originally developed by Ondřej Klejch, Joachim Fainberg and Joanna Rownicka, from the CSTR group in Edinburgh University, for their participation in the Kaggle challenge mentioned earlier. They themselves started from the recipe[4] provided with Kaldi for the Resource Management (RM) dataset.

The CNN-BiLSTM acoustic model is composed of two 2D (across both time and frequency) convolutional layers, followed by 4 fully connected layers. The network is trained to predict senones with the usual cross-entropy loss, followed by 1 iteration of discriminative training with the sMBR objective (Gibson and Hain, 2006). The network uses as input 11 frames (with a context of 5 from both sides) of 40 filter bank coefficients, augmented with Δ and $\Delta\Delta$ features. The alignments used as reference for network optimization are obtained from a GMM-HMM system trained with LDA-MLLT features.

All the words from the `_unknown_` category are modeled through a single occurence of a special phoneme labelled `UNK` (e.g. *house* is transcribed as `UNK`). For decoding, a grammar allowing a single occurrence of any of the 10 keywords or the `UNK`/`SIL` phonemes (in isolation) is used.

Table 5.1 compares the performance of this model and the classifier provided by Tensorflow team on version 1 of the dataset. It can be seen that the CNN-HMM system clearly outperforms the classifier, with 74% of relative improvement on the classification error for the test set. The CNN-HMM will thus be retained for comparison in following experiments.

[4] `egs/rm/s5/local/nnet/run_cnn2d.sh`

5.4.3 End-to-end model

I report here experiments on hybrid CTC/Attention with either pyramidal or CNN encoders. The pyramidal BiLSTM encoder is composed of 4 layers of 320 bidirectional LSTM units. Downsampling is performed in the first two layers with a factor of 2 each time. The CNN-BiLSTM encoder is composed of 4 convolutional layers, with 2 max pooling layers (after the second and fourth convolutions). Each pooling layer has a reduction factor of 2, thus downsampling the timescale of the input by 4 overall. Four layers of 320 BiLSTM units sit on top of the CNN part.

I used the location-aware attention mechanism and a layer of 300 LSTM cells for the decoder. Default hyperparameters from ESPnet `voxforge` recipe were used unless stated otherwise, including the default value of $\lambda = 0.5$. The input was composed of 80 fbanks and I experimented with 3 different types of labels: phonemes, graphemes and words.

When a phoneme-based output is used, each keyword is mapped to a phoneme sequence using a pronunciation dictionary. Also, similarly to what is done for the CNN-HMM baseline, the additional `UNK` and `SIL` phonemes are used to model examples from the `_unknown_` and `_silence_` categories (each example being modeled with a single occurrence of the corresponding phoneme). For the grapheme-based output, each keyword is decomposed in its constituent character sequence while `?` and `_` characters are used to represent the `_unknown_` and `_silence_` categories (e.g *house* is transcribed as `?`). For both phoneme- and grapheme-based outputs, all output not corresponding to one of the keywords or `_silence_` is further mapped to the `_unknown_` category when scoring.

5.4.4 Strategies for few-shot learning

The main limitation of the small vocabulary approach used here (or used in the Kaggle challenge) is its flexibility. The ASR systems presented here are trained on a small set of keywords and no guarantee is given that they will generalize to new ones (in fact I expect them to recognize new keywords poorly if at all). While the use of sub-word targets usually allows to decode words not seen at training time, the poor coverage of senones (due to the limited vocabulary) is likely to result in poor generalization to new keywords. Also, in the case of the S2S model, the decoder may overfit to the small vocabulary and thus be unable to predict any other sequence. This is a limitation that is hardly manageable in practical usage. To alleviate it, I propose to explore strategies for few-shot learning, where one can gather few examples of a new word and use them to retrain or adapt the existing system, so that it will perform better on this new word.

The `retrain` strategy. The simplest strategy I tried consists in adding the examples of the new keywords to the training set from the beginning and train a new model on it (a method referred to as `retrain` hereafter). One issue with this method is that the new keywords will be under-represented compared to the original ones. To improve on that, I propose to try oversampling the few-shot examples, that is the model will see these examples k times during an epoch (where k is the oversampling factor) when the original examples will be seen only once.

The `adapt` strategy. The main limitation of the `retrain` strategy is that it requires to retrain the model from scratch every time. Alternatively, I propose a method based on adaptation (referred to as `adapt`) where I start from a model trained on the 12 original categories (the 10 original keywords to which `_unknown_` and `_silence_` categories are added). Its weights are then adapted by training the model for a few more epochs on the few-shot examples, keeping the same training procedure otherwise. To avoid performance deteriorating too much on the 12 original categories, some of their examples are also included, with the same number of examples per class as for the few-shot classes. The overall number of examples being very small, few updates are made per epoch. I thus expect higher learning rates to be useful. The number of epochs, which plays a complementary role, has also been optimized.

One drawback of this strategy, however, is that the model I start from may not contain all the output labels required for the new keywords (limited to the phonemes or graphemes present in the 10 original keywords). This problem is solved by replacing the missing phonemes (resp. graphemes) by the `UNK` model (resp. the character `?`) initially introduced for the `_unknown_` category). This can result in a dramatic change as exemplified by the word *backward* for which most phonemes are absent from the pretrained models output. The original transcription `"B AE K W ER D"` becomes `"UNK UNK UNK UNK UNK D"` after replacement. Similarly for graphemes, replacing missing characters leads to the transcription `"????w?rd"`.

The `retrain_replace` strategy. In view of the limitation just mentioned and in order to compare the `adapt` strategy more fairly with the `retrain` approach (which uses all labels where the `adapt` strategy do not), the `retrain_replace` strategy is introduced. This strategy uses the same training procedure as the `retrain` method, but with the modified labels (limited to the phonemes/graphemes present in the original keywords).

TABLE 5.2 List of SC keywords assigned to the different word sets.

Set	Words
`org_kwd`	*down, go, left, no, off, on, right, stop, up, yes*
`org_unk`	*bed, bird, cat, dog, happy, house, marvin, sheila, tree, visual, wow*
`new_kwd`	*forward, four, one, three, two, zero*
`new_unk`	*eight, five, follow, learn, nine, seven, six*

5.5 Experiments

5.5.1 Experimental setup

My experimental setup is based on the second version of the SC dataset presented in section 3.2.4. The original setup with 10 keywords has been slightly modified here. A limitation of the original design is that the same collection of words is used for the `_unknown_` category at both training and test time. In order to better evaluate the generalization capability of the model to unseen words (which will likely be limited given the number of words used to train this category), I decided to exclude some of them from the training set. These words will only be used for evaluation in order to get a better idea of the generalization of the `_unknown_` category to unseen speech. Also, as mentioned earlier, I am interested in exploring strategies for few-shot learning. Hence, a few words are also kept aside for use in those experiments.

The 10 original keywords (from here on referred to as the `org_kwd` set of words) are kept identical. The 25 remaining ones however are split into two main categories: 7 are used as new keywords in the few-shot experiments (referred to as `new_kwd`) and 18 as unknowns (the `unk` set). This later group is further split into 11 words (`org_unk`) that are used for training and evaluation, while the remaining 7 (`new_unk`) are seen at evaluation time only. When evaluating the `unk` set, the average of the scores of the two subsets is used (with equal weight for the two categories). Table 5.2 gives the list of words assigned to each category.

The split of the data in training, validation and test sets was done using the function implemented in Tensorflow's example code with 80%, 10% and 10% for each set respectively. The `_unknown_` category being the combination of several keywords, it is over-represented in the dataset. To prevent it from dominating the learning procedure, it is downsampled by randomly selecting a number of examples corresponding to the mean number of examples available for the keywords (`org_kwd`). Finally, for all experiments on few-shot learning, f examples are randomly sampled from the training records I have kept aside for each new class.

TABLE 5.3 Error rate (%) of the S2S model on SC validation set, with different types of encoder and output. Unlike Table 5.1, results for the 10 keywords and the `_unknown_` or `_silence_` categories are given separately.

Encoder	**Output**	`org_kwd`	`unk`	`_silence_`
Pyramidal BiLSTM	Phoneme-based	**4.9**	27.4	0.0
	Grapheme-based	5.1	**25.4**	0.0
	Word-based	5.5	31.1	0.0
CNN-BiLSTM	Phoneme-based	**3.7**	**22.2**	0.0
	Grapheme-based	3.8	23.1	0.0
	Word-based	3.8	25.7	0.0

5.5.2 End-to-end approach for small vocabulary ASR

I first report results on the original classification task trained on 12 categories, comparing traditional and E2E pipelines. Table 5.3 summarizes the results obtained with the S2S model for different types of encoder and output. Unlike Table 5.1, error rates for the 10 keywords (`org_kwd` set) and the `_unknown_` or `_silence_` categories are given separately. We can first see that the encoder using CNNs clearly outperforms the one based on pyramidal BiLSTMs. The best CNN-BiLSTM model reduces the classification error by 24% relatively to the best pyramidal model on the `org_kwd` category. Comparing the different types of output now, we see that the classification error of the three CNN-BiLSTM S2S models on the keywords are very close. Looking at the performance of the same models on the `unk` set, we see they all display a much higher error rate, as expected with only 11 different words to populate the `_unknown_` category for training. It can be noticed though that the phoneme-based S2S model performs significantly better than its two competitors on this set. This may be due to a better generalization to the `new_kwd` set which counts for half in the error rate of the `unk` category.

Hence, the greater simplicity of the grapheme- or word-based approaches, which do not require a pronunciation dictionary, can be traded off for better performance. Moreover, it has to be highlighted that in the small vocabulary context explored here, building the pronunciation dictionary is greatly simplified compared to the large vocabulary situation.

In Table 5.4, I compare the test classification error of the best E2E model (CNN-BiLSTM encoder with phoneme-based output) with the CNN-HMM baseline. On the main task, the E2E approach beats the baseline by 40% relative. This is very promising as it shows that E2E models are a competitive alternative to more traditional approaches for this task. The

TABLE 5.4 Classification error (%) of the baseline and the best S2S model on SC test set.

Model	`org_kwd`	`unk`	`_silence_`
CNN-HMM	4.2	35.6	0.0
Hybrid CTC/Attention S2S	**2.5**	**23.6**	0.0

results on the `unk` set shows that they also generalize much better, the E2E approach beating the baseline by 34% relative on this subset.

Finally, I give some insight on the behavior of the S2S models. As can be seen from Figure 5.2, and confirmed by manual inspection, the attention tends to focus on a single portion of each input or (more often) uses a diffuse attention that spans most of the input. Contrary to what is usually observed with this kind of architecture, the attention does not shift along the input time axis as the output tokens are produced. It appears that the model representation is more akin to word than sub-word modeling, as is usually observed. With the small vocabulary used here, the model is apparently able to discriminate between the different keywords with a single "glance" at the data. For example, in the case of the word *stop*, the model seems to attend to the phoneme `T` (first row of Figure 5.2). More surprisingly, in some cases (a good example is the word *yes*, 2^{nd} row of the figure), the model seems to seek information from a fixed position, even if it falls in the silence preceding the word. The most common case is illustrated by the remaining examples where the attention is spread over the input. A more quantitative analysis would be required to better understand those dynamics, which may be related to the effective window size of the encoder.

5.5.3 Few-shot learning

For small number of few-shot examples (f), the variability introduced by the random selection of the sample is high. Mean scores over 10 runs is thus reported for all experiments on few-shot learning.

I experiment with $f \in \{10, 100\}$. While 100 examples may seem a lot for few-shot learning, it allows to test how the different strategies behave when the number of examples increase. It is also to be noted that 100 examples is only 2.6% of the number of examples available for the original classes ($\sim$ 3850 on average).

For the `retrain` strategy, I experimented with values of the oversampling factor (k) so as to reach up to 3000 simulated examples for the new keywords ($k = 300$ for $f = 10$ and $k = 30$ for $f = 100$), obtaining similar frequency in training and test sets. As Figure 5.3 shows,

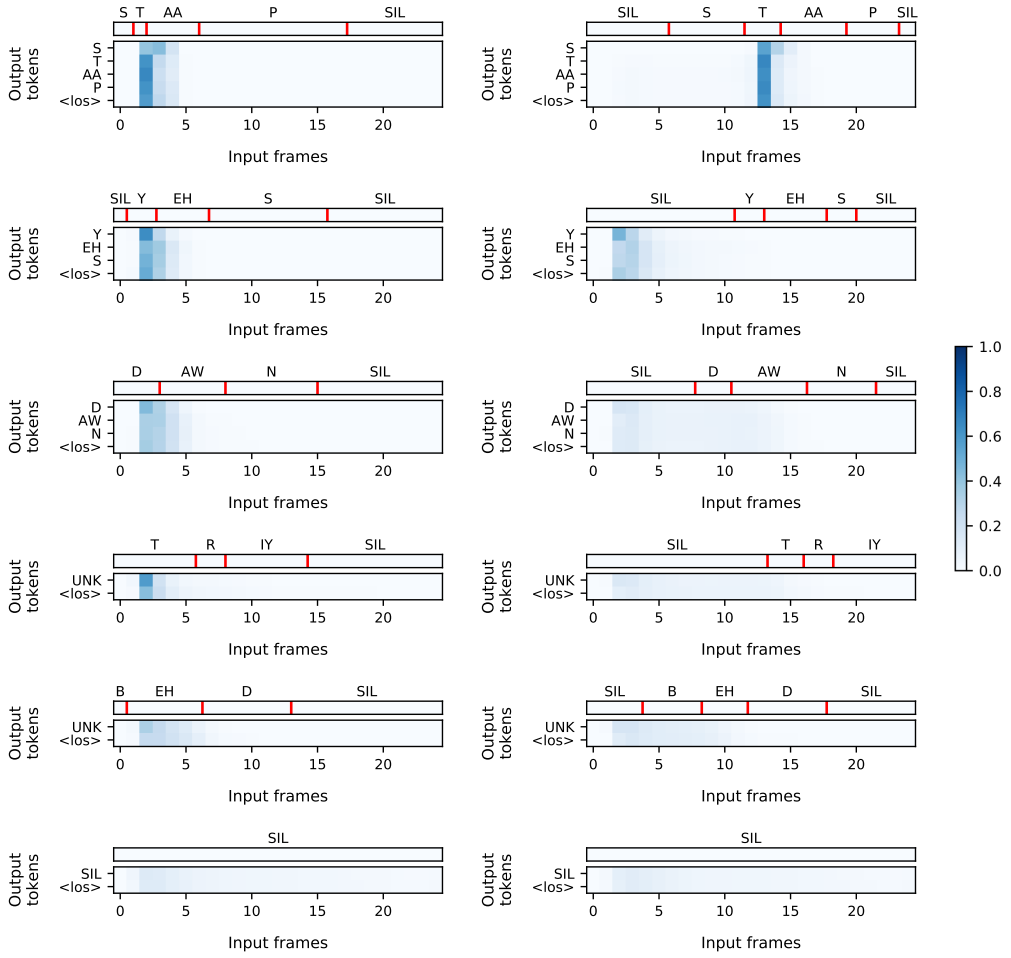

FIGURE 5.2 Attention weights produced by the phoneme-based S2S model for two examples of the following words (one line per word in the same order): *stop*, *yes*, *down*, *tree* (unknown), *bed* (unknown) and silence. Their respective alignment is displayed on top.

this technique is particularly effective for $f = 10$ with the `new_unk` set (top figure). While the score of both phoneme- and grapheme-based models is around 95% of error without oversampling, the mean error score goes below 72% when the multiplication factor is higher than 1 and below 60% for multiplication factors between 10 and 100. The best score is achieved for $k = 10$ with the grapheme-based output, which improves the performance by 40% relative over the same model without oversampling. The improvement is less visible for $f = 100$ but still lead to 12% relative improvement for the grapheme-based model with $k = 30$.

The degradation of the performance on the original keywords (middle figure) tends to increase with the multiplication factor, peaking at $k = 100$ for $f = 10$ and then going down. The degradation for the two models giving the best results on the new keywords is around 16% in both cases. A better compromise between new and original keywords can probably be found though. Finally, we can see that the performance of the different models on the `unk` set (bottom figure) is quite variable with no clear tendency. The mean classification error fluctuates between 24 and 31% for $f = 10$ and between 27 and 31% for $f = 100$.

Figure 5.4 shows the classification error with the `retrain_replace` strategy for the `new_kwd` (top), `org_kwd` (middle) and `unk` (bottom) categories. As can be seen from the figures, the phoneme replacement rule does not seem to affect considerably the results. For $f = 10$, the best results on the new keywords are achieved with the grapheme-based model and $k = 10$, with an error rate of 57.9%. Conversely for $f = 100$, the best performance is obtained with the phoneme-based model and $k = 30$, with a classification error of 17.0%. The main difference between the `retrain` and `retrain_replace` strategies is in the results obtained for the `unk` category which are much more consistent here. The scores obtained with phoneme- and grapheme- based outputs with the later strategy are more correlated for both values of f. These results tend to suggest that the inclusion of new phonemes/graphemes in the `retrain` strategy may be the reason of the highly variable performance on the `unk` set.

Finally, Figure 5.5 shows the classification error with the `adapt` strategy for the `new_kwd` (top), `org_kwd` (middle) and `unk` (bottom) sets. For each experiment, the best number of epochs has been selected based on the validation set. As can be seen from the top figure, this strategy allows to reach much lower error rates on the new keywords. We can also see that increasing the learning rate is necessary to get the best results, but too high a value and the results quickly deteriorate as the model overfits the small training set. The best results on the new keywords are achieved with the grapheme-based (resp. phoneme-based) output, with a learning rate of 3 for $f = 10$ (resp. $f = 100$). The performance on the `org_kwd` set

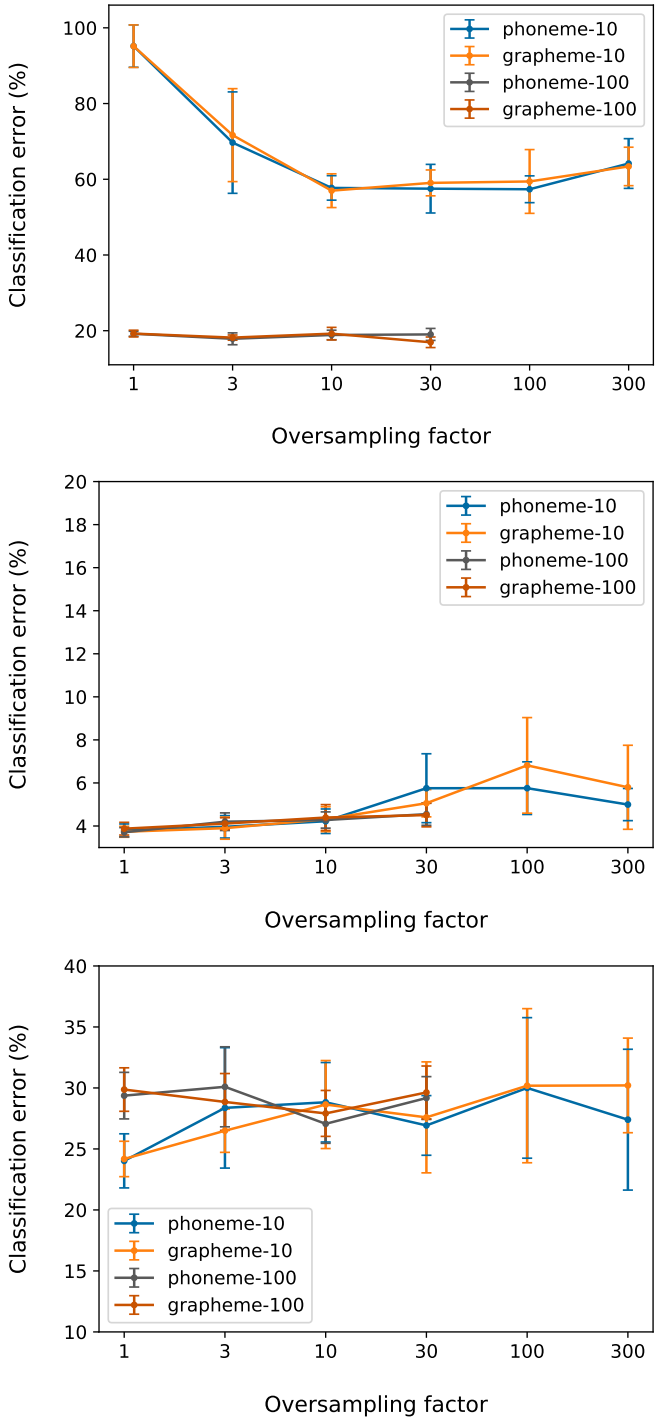

FIGURE 5.3 Classification error (%) on SC validation set using the `retrain` strategy, for the `new_kwd` (top), `org_kwd` (middle) and `unk` (bottom) categories and as a function of the oversampling factor. Phoneme- and grapheme-based outputs are compared, for $f = 10$ or 100. The bars represent the standard deviation over 10 runs.

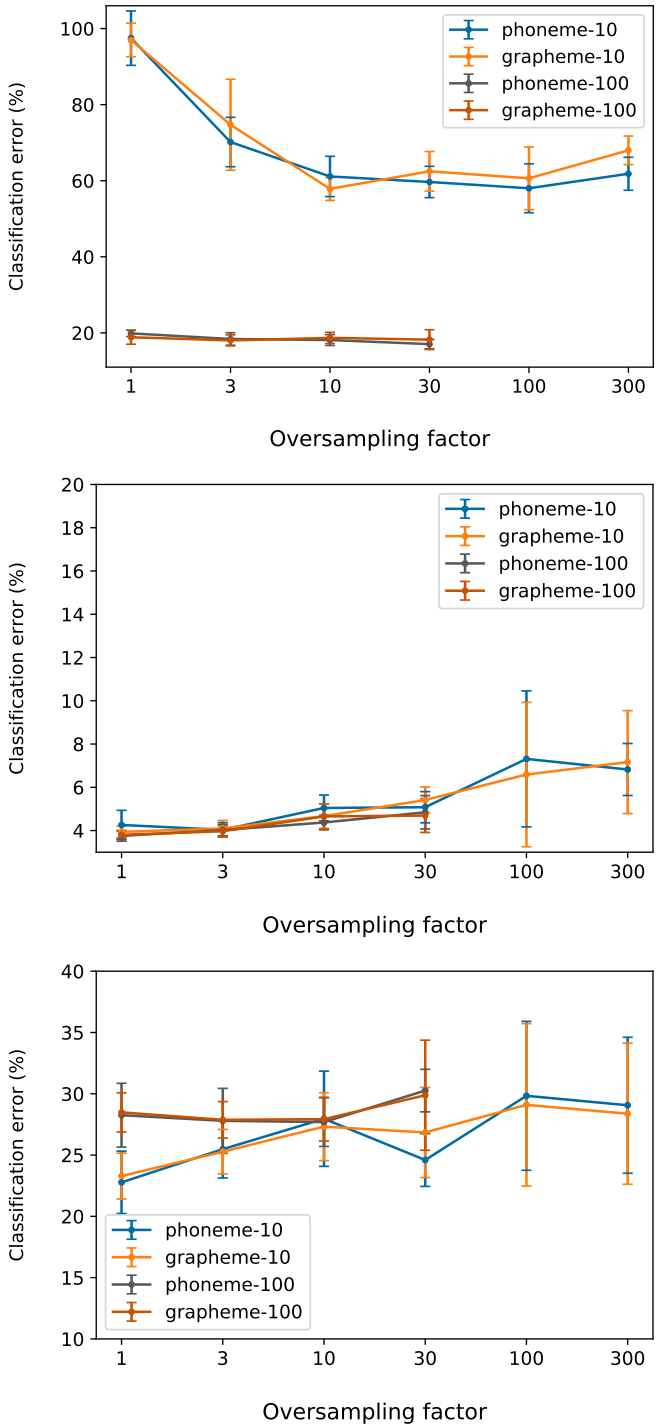

FIGURE 5.4 Classification error (%) on SC validation set using the `retrain_replace` strategy, for the `new_kwd` (top), `org_kwd` (middle) and `unk` (bottom) categories and as a function of the oversampling factor. Phoneme- and grapheme-based outputs are compared, for $f = 10$ or 100. The bars represent the standard deviation over 10 runs.

TABLE 5.5 Error rate (%) on SC validation set using the S2S model with different few-shot learning strategies and different number of few-shot examples.

f	Strategy	org_kwd	unk	_silence_	new_kwd
10	retrain	**4.3**	28.6	**0.1**	57.0
	retrain_replace	4.7	**27.3**	**0.1**	57.9
	adapt	11.5	65.8	0.3	**29.2**
100	retrain	**4.5**	**29.6**	**0.0**	16.9
	retrain_replace	4.8	30.3	**0.0**	17.0
	adapt	7.4	42.5	0.5	**10.1**

TABLE 5.6 Error rate (%) on SC test set using the S2S model with the two main few-shot learning strategies.

f	Strategy	org_kwd	unk	_silence_	new_kwd
10	retrain	**3.7**	**31.3**	**0.1**	59.5
	adapt	11.7	66.5	0.4	**31.2**
100	retrain	**3.8**	**31.8**	0.0	18.5
	adapt	6.9	45.0	0.8	**11.6**

progressively deteriorates as the learning rate increases and a big drop in performance is observed for a learning rate of 7.

Table 5.5 summarizes the error rate of the different strategies on the validation set for the two values of f (10 and 100) with the hyperparameters giving optimal scores on the `new_kwd` set. A first observation is that the phoneme/grapheme replacement rules introduced in subsection 5.4.4 for the `adapt` strategy does not seem to penalize the performance on the `new_kwd` set. The `retrain_replace` strategy gives results very close to the `retrain` one overall. Comparing the performance of the `adapt` and `retrain` strategies now, we see that adaption is not only much faster to train but is also the best one on the new keywords. The classification error is improved by 49% and 40% relative for f 10 and 100. Though, this is is achieved at the expense of the `org_kwd` and `unk` sets where the performance is highly deteriorated. Lower learning rates may help mitigate this issue by providing a better compromise between the new keywords and the other categories.

Table 5.6 summarizes the test error rate of the best models for both strategies (`retrain` and `adapt`).

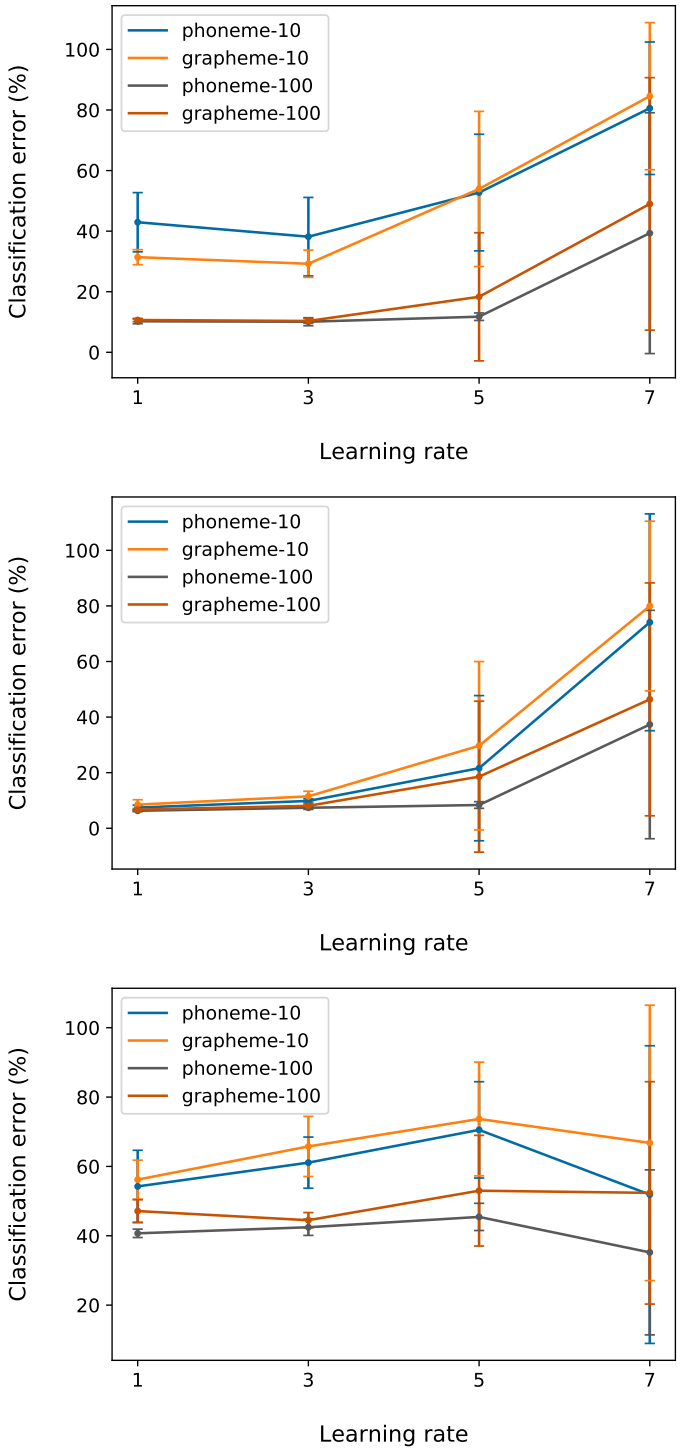

FIGURE 5.5 Classification error (%) on SC validation set using the `adapt` strategy, for the `new_kwd` (top), `org_kwd` (middle) and `unk` (bottom) categories and as a function of the learning rate. Phoneme- and grapheme-based outputs are compared for 10 and 100 fewshot-examples (f). The bars represent the standard deviation over 10 runs.

5.6 Conclusion

In this chapter, I studied the adequacy of E2E approaches on a small vocabulary task, in order to simplify the process of training a keyword/command recognition system and make this technology more accessible. I found that they can be competitive in such a context, giving better results than a strong CNN-HMM baseline. Two few-shot strategies have also been proposed. By simply training a model from scratch on the combination of the original dataset and those new examples, I managed to reach 40.5% of accuracy with only 10 examples per new keyword and 81.5% with 100 examples. A faster adaptation strategy was also proposed which achieves even better results with 68.8% and 88.4% of accuracy for 10 and 100 examples respectively. Though, this is achieved at the expense of the performance on the original keywords.

I have also shown that the dynamic of the hybrid CTC/Attention model in the proposed task is quite different from what is usually observed with large vocabulary tasks. The model representation is more akin to word modeling making the attention mechanism of limited interest. An alternative then would be to consider a many-to-one model such as the attention-based system (without decoder) proposed by Shan et al. (2018). It would also be interesting in the future to analyze more deeply the behavior of the model as one move from a small vocabulary keyword task to a large vocabulary one with complex sentences.

Chapter 6

Integration

The ultimate goal of my research is the study and development of command recognition technology for robotics. In parallel to my work on acoustic model robustness and few-shot learning for sequence-to-sequence models, I developed a fully functional command recognition system for the iCub platform[1] (Metta, Natale, et al., 2010), called iCubrec. This system is intended as a testbed for the proposed methodologies and a demonstration of their performance on a real application case.

A first version of this system[2] as been described in Higy, Mereta, et al. (2018). The speech recognition system was then based on Tensorflow and HTK. A new version has since been developed, which improves the pipeline in several ways:

- Segmentation of speech from the continuous audio stream is done automatically by a VAD module, instead of the manual trigger used previously.
- HTK toolkit has been replaced with Kaldi which offers more facilities for neural networks integration, as well as many modern techniques not available in HTK.
- A stronger DNN-based acoustic model has been trained.
- A garbage model has been added to the grammar of our model, as well as a rejection mechanism to account for failures of the VAD model.
- A simple mechanism has been implemented to filter out sentences pronounced by the robot itself.

I will briefly present the iCub platform, before describing the developed speech recognition pipeline in more details.

[1] http://www.icub.org/index.php
[2] https://github.com/robotology/natural-speech/tree/master/icubrec

6.1 iCub platform

iCub is an open-source humanoid robotic platform, intended for research in embodied artificial intelligence. The robot was originally designed by a consortium of several european universities, in the scope of the RobotCub european project, and built by the Italian Institute of Technology (IIT). In addition to the robot itself, the platform also includes a middleware and a large software repository. The middleware, called YARP (Yet Another Robot Platform) (Metta, Fitzpatrick, et al., 2006), is responsible for the communication between the sensors, the actuators and the different modules controlling the behavior of the robot. The module repository[3], mainly written in C++, provides many basic functionalities (e.g. visual depth perception, motor control, voice synthesis, facial expressions) as well as more advanced applications (e.g. visual object detection and recognition, walking and balance control, object grasping and manipulation). The hardware design, the software and the documentation are all released in open-source under the GPL license.

6.1.1 The iCub robot

The iCub (illustrated in Figure 6.1), which is 104 cm tall and weights around 22 kg, has been dimensioned as a five-year-old child. It has 53 degrees of freedom (DOF) divided between the head, arms and hands, waist and legs. In addition to visual and audio perceptions, the robot has full kinesthetic awareness: proprioception (body configuration in space), movement (through accelerometers and gyroscopes), force/torque (to control how much force is exerted) and touch. It is one of the few humanoids in the world with a full-body sensitive skin allowing it to interact with its environment safely.

The robot was not initially designed to operate autonomously and was provided power and network connectivity through an umbilical cable. A battery-packed version is now available as well (as of version 2.5), with wifi connection and on-board battery (giving the robot an autonomy of 2 hours).

More than 40 robots exist in various laboratories world-wide.

6.1.2 R1

R1 (illustrated in Figure 6.2) is a new robot developed by the IIT. Intended as a personal assistant for domestic or professional purposes, it is 125 cm tall and weights around 50 kg (including a battery, which gives it an autonomy of 3 hours). Its extensible torso and arms

[3] https://github.com/robotology

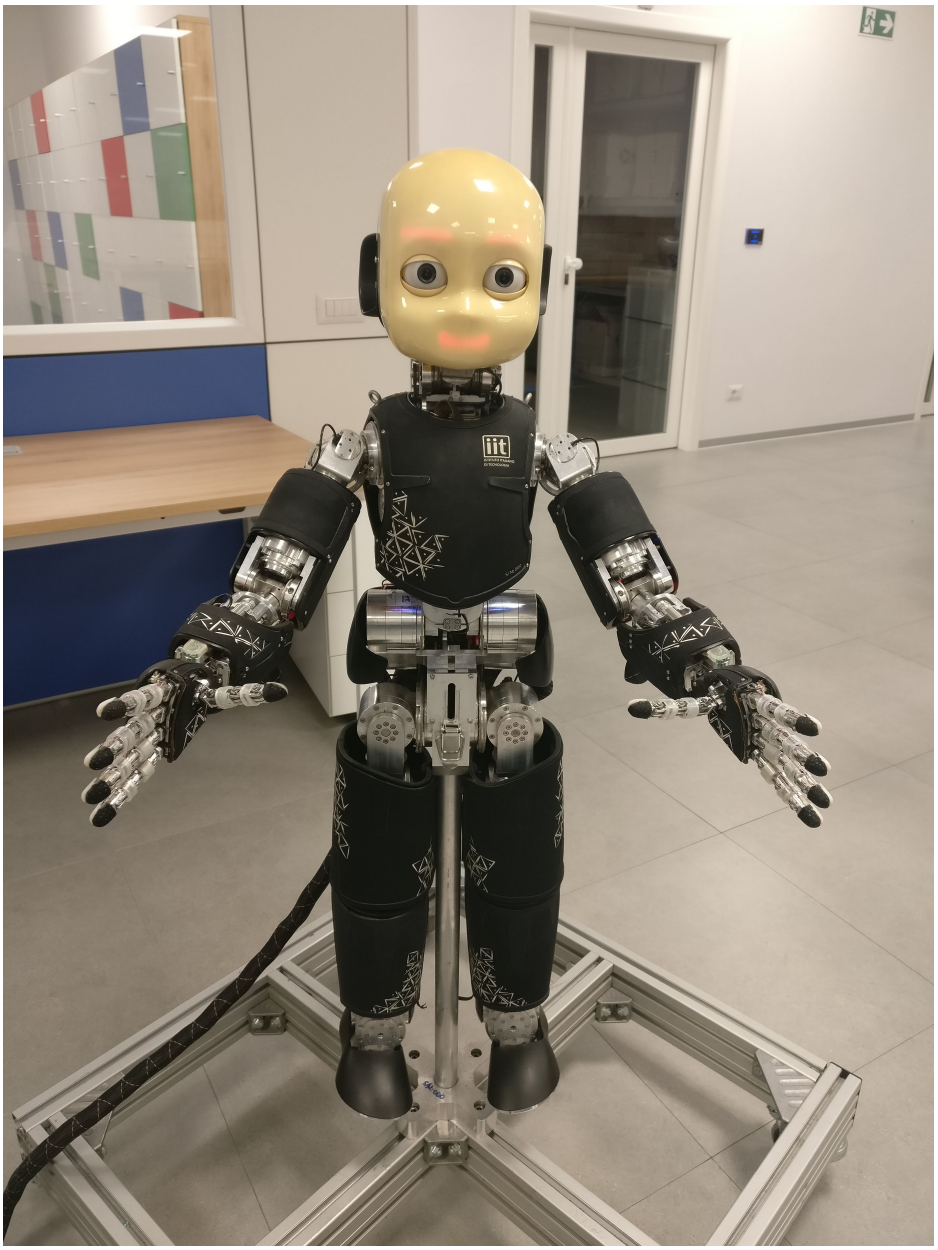

FIGURE 6.1 iCub robot.

allow the robot to reach far objects. The head, composed of a LED screen, also contains vision, equilibrium and sound sensors.

Aimed at mass market, its design has been simplified compared to iCub, in order to make it more robust and affordable. Hence, it is composed at 50% of plastic and 50% of carbon fibers and metal. The number of motors as been reduced from 53 to 28. In particular, the hands are composed of two fingers only, the legs are replaced by gyroscopic wheels and the tactile skin is present only on the forearms and the hand.

6.1.3 YARP

Yet Another Robot Platform (YARP) is the middleware developed for the iCub platform, which purpose is similar to the well-known Robot Operating System (ROS) platform which came after it. It takes care of the communication between sensors, actuators and the diverse modules that control the robot. While YARP is originally developed in C++, it can be interfaced with other programming languages like Python. As all other components of the iCub platform, it is open sourced (under BSD-3-Clause license). Performance plays a key role in the design of YARP.

Ports are a key concept in YARP, as they allow the different modules to exchange information. For example, they are used to handle communication between the different modules of the iCubrec command recognition system. Each module can declare any number of ports which will be used to send or receive information. The port's name, serves as an id to refer to the port from outside the module and can for instance be used to connect two ports. Whenever data is written on an output port, the port will send it through YARP to all ports registered as listeners. A simple example is provided in Figure 6.3 with two modules, a sender with an output port named `/sender:o` and a receiver with an input port named `/receiver:i`. The two ports are connected such that any data sent by the sender on its output port will automatically be received by the receiver on its input port. The suffixes `:o` and `:i` are not mandatory but are an easy way to identify input and output ports. I will follow this convention here.

Remote procedure call (RPC) ports are also available which allow synchronous communication. Hence, each message sent will receive a reply. This mechanism can be used for example when the sender needs to receive a confirmation that the action has been performed. Following the convention introduced previously for ports, RPC ports will be prefixed with `:io`.

FIGURE 6.2 R1 robot.

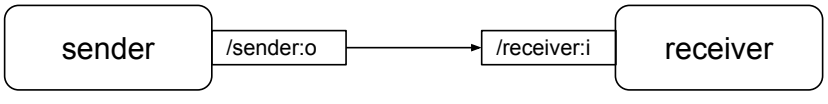

FIGURE 6.3 An example of two modules connected through YARP ports.

FIGURE 6.4 Placement of the microphones on iCub's head, one on each side of the head, at a position similar to human ears. The right microphone is represented here, indicated by the blue arrow.

6.1.4 Speech recognition on the iCub platform

Two main options are available to acquire audio when working with the robots: (1) use the robots' own microphones, or (2) use an external microphone. Both R1 and iCub are equipped with microphones. iCub has two microphones, one on each side of the head, at a position similar to human ears (see Figure 6.4). R1, is equipped with an array of 6 microphones, all mounted on the head (see Figure 6.5). At the beginning of my PhD, R1 was not yet available and the version of iCub we had (version 2.5) was not ideal for speech recognition. First, the microphones available on the robot were of poor quality with a low SNR. Second, one inconvenience of iCub, as far as audio perception is concerned, is the presence of the on-board PC104 controller in the head of the robot. The fan used to cool the system is quite noisy and very close to the microphones. An alternative has been found which relies on an external microphone for sound acquisition (used to gather VoCub dataset). These problems have now been solved with a new version of the head incorporating better microphones (Soundman OKM II) and a more silent fan-less cooling system.

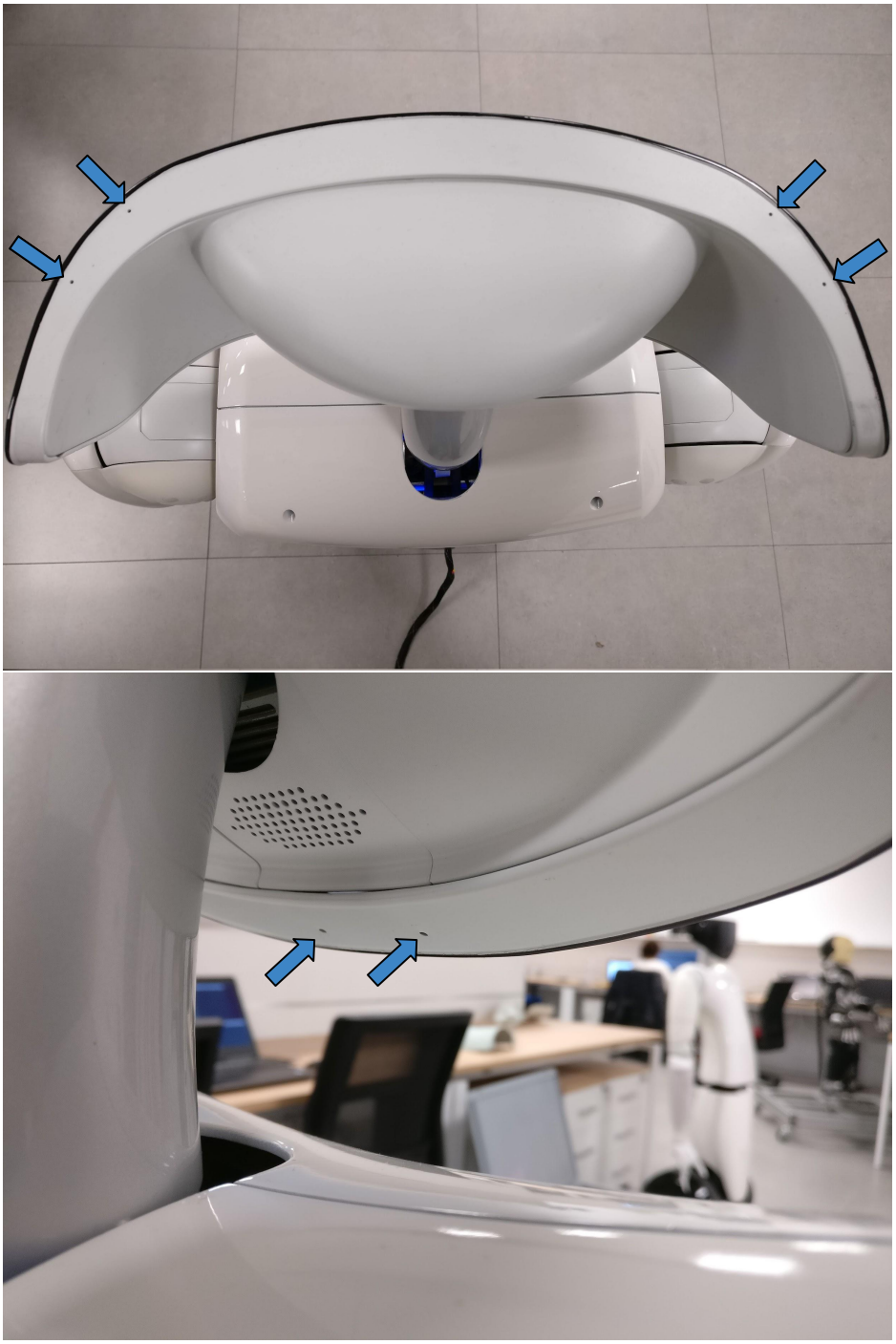

FIGURE 6.5 Placement of the microphones on R1's head, 4 on the top (top picture) and 2 on the bottom of the head(bottom picture). The location of the microphones is materialized by the small holes indicated by the blue arrows.

The alternative using an external microphone is based on the yarp.js[4] library (Ciliberto, 2017). The library uses javascript, and more precisely Node.js, to connect any device to YARP without requiring any code installed on the machine itself. A web server, responsible for the communication with YARP offers web services that can be accessed by any device on the same network through a web browser. This way, it is possible to acquire sound from a phone and stream it through YARP for example. This is the method that was used to record the VoCub dataset from a tablet.

It is to be noted that YARP can handle sounds natively through the `Sound` object.

6.2 iCubrec pipeline

The command recognition pipeline we developed is organized as follows:

- The sound is acquired from one of the robots microphones or from an external device and streamed through YARP.
- The VAD system detects the presence of speech in the continuous audio stream and segments the relevant portion. The segment corresponding to a possible command is saved to a file and the actual command recognition system is activated.
- The command recognition module decodes the sentence. A garbage model and a rejection mechanism allow the module to filter false positives from the VAD (see subsection 6.2.2 for details). When a command is recognized, it is sent to a state-machine.
- The state-machine module is responsible for controlling the interaction with the robot, generating appropriate actions in response to the commands, based on the context and the state of the interaction.

The pipeline is illustrated in Figure 6.6.

The sound acquisition process and the state-machine where not the focal point of my work and mainly rely on pre-existing modules. I mostly contributed to the implementation of the VAD and command recognition systems, which I will now describe. More focus is put on the command recognition module and especially the acoustic model which are the main focus of my thesis.

[4] https://github.com/robotology/yarp.js

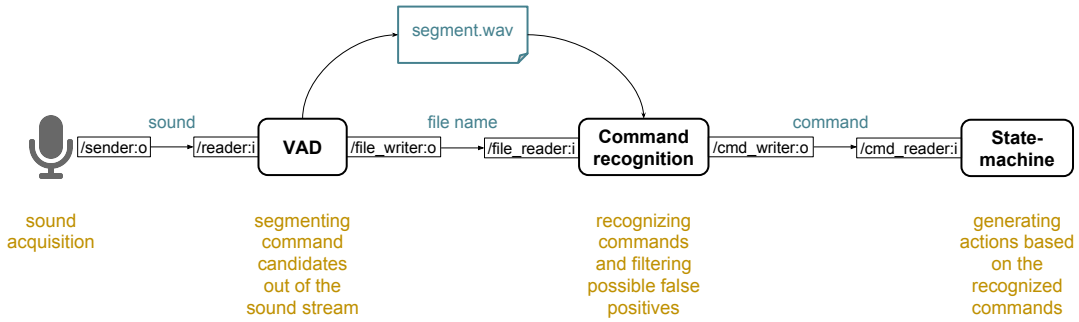

FIGURE 6.6 The command recognition pipeline.

6.2.1 Voice activity detection

The development of the VAD system is a joint work of 3 persons: Raffaele Tavarone (Center for Translational Neurophysiology of Speech and Communication (CTNSC), IIT), Luca Pasa (same affiliation) and I. Raffaele Tavarone focused on the training of a DNN-based VAD model, while together with Luca Pasa, I worked on the integration of this model in YARP.

To construct the neural network input, 40 log mel-scale filter bank coefficients and energy are extracted over 25 ms frames, with a stride of 10 ms. We then concatenate the features from 11 frames (5 from each side of the current frame) to construct the final input. The network is composed of 4 layers of 2000 units, with the ReLU activation function for the hidden layers and the softmax function for the output one. The output has a dimension of 2 corresponding to the probabilities of "speech" versus "non-speech". The network is trained on VoCub dataset with the standard cross-entropy loss, through SGD and with mini-batches of size 1500. Adam optimizer (Kingma and Ba, 2015) is used with a learning rate of 0.0001 and $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\varepsilon = 1e^{-08}$.

Figure 6.7 shows the confusion matrix for the frame-level predictions of the VAD on VoCub test set and for the chosen operating point. The accuracy for the non-speech and speech categories is of 96% and 92% respectively. Evaluation of a similar model trained on a different dataset can be found in Savran et al. (2018).

The output of the VAD is further smoothed using a moving average over 5 outputs (2 on each side of the current one). Speech is detected when the probability exceeds a threshold (0.7 by default). A minimum length parameter (set to 300 ms by default) also allows to filter out segments that are unlikely to contain speech due to their shortness.

The neural network is trained and runs with Tensorflow. The module is interfaced with YARP through its Python bindings. When a segment of speech is detected that exceeds

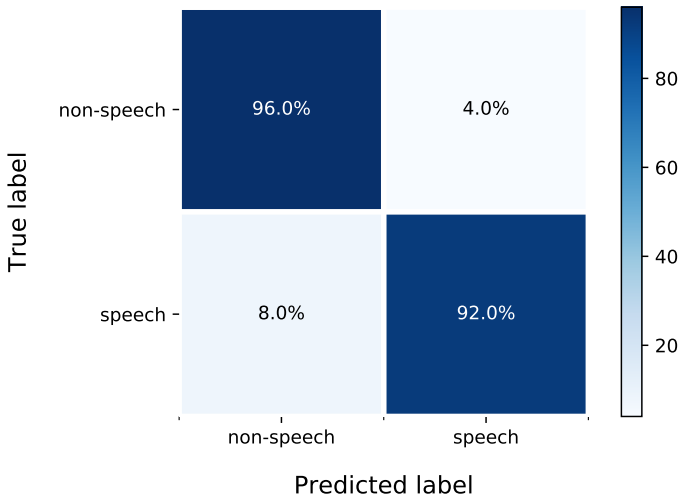

FIGURE 6.7 Confusion matrix for the frame-level predictions of the VAD on VoCub test set.

this minimum duration, it is saved as a wavefile and the command recognition module is notified. The module exposes two ports: `/reader:i` where it receives the incoming sound, and `/file_writer:o` where segment filenames are written upon detection.

Hyperparameters where chosen so as to favor false positives over false negatives. This allows to limit the number of commands missed by the VAD system, while the false detections can be subsequently filtered by the command recognition system.

6.2.2 Command recognition

The first version of the command recognition system, described in Higy, Mereta, et al. (2018), is a GMM-HMM system based on HTK. In order to interface it with YARP, the original `HVite` decoder (written in C) has been modified[5] to add two ports. Speech recognition is triggered by reception of the command `"recognize"` on the input port, while the output port is used to sent the recognized command (in textual format). One of the main limitation of using HTK, apart from the limited support for neural networks mentioned in subsection 3.1.1, is the impossibility to use neural networks for online decoding. Only GMM-based models are supported (as of version 3.5). This is another reason that motivated the use of Kaldi for the second version.

[5]The modified version of HVite decoder can be found in the github repository containing all the code for the first version of iCubrec: https://github.com/robotology/natural-speech/tree/master/icubrec/

While I implemented the first version of the command recognition system by myself, the second version is a joint work of Leonardo Badino, Luca Pasa and myself. The DNN-based acoustic model was trained by Leonardo Badino and will be presented in more detail in next section. I worked with Luca Pasa on the integration of the model with YARP. For each segment extracted by the VAD, senones posterior probabilities are estimated by the DNN. Similarly to the VAD system, Tensorflow is used to train and run the acoustic model. Once extracted, the posterior probabilities are passed to Kaldi for decoding and the best lattice path is extracted. A finite state transducer based on VoCub grammar (see section A.1) is used as language model. The Python module is interfaced with YARP through two ports: `/file_reader:o` where the name of the wavefile containing the segment to decode is expected, and `/cmd_writer:o` where the recognized command is sent (in textual format).

Even though the VAD system should filter most of the non-speech audio, it may still generate some false positives (especially so as we favor them over false negatives). We combine two mechanism to filter them: (i) a filler model inspired by literature on KWS, and (ii) a rejection mechanism based on a confidence score.

The filler model was initially composed of a special phoneme (`<SPN>`) modeling non-command speech noise. However, we noticed many false positives due to non-command sentences containing words present in the commands (especially stop words such as *I*, *you*, *is*, or *this*). We thus included the most frequent stop words[6] in the garbage model, in isolation. This heuristic proved very effective in filtering out non-command sentences.

Additionally, a filtering mechanism based on the per-frame average of the acoustic negative log-likelihood (given by Kaldi program `lattice-best-path`) allows to remove segments that are likely to be mis-recognized given their high score. A threshold of 1.7 gives good results in practice.

Finally, an unanticipated source of errors came from the robot. Speech synthesis is used on the robot to interact with the user. In the course of an interaction, the robot often pronounces sentences very similar to the commands used by the human (if not the same). An example would be the robot saying *"This is an octopus"* to answer the question *"What is this?"*, which is exactly one of the commands. While more advanced mechanisms could be used, we simply disregard audio while the robot is speaking.

[6]The complete list is the following: *I*, *is*, *no*, *see*, *this*, *yes*, *you*.

6.3 The acoustic model – current performance and limitations

I will now present the hybrid DNN-HMM acoustic model in more details with an evaluation of its performance. In all cases, the command recognition system is evaluated in isolation here.

In order to improve on the baseline presented in section 3.2.5, we explored two domain adaptation strategies: (1) multi-condition training (referred to as `multi-condition` hereafter), where one uses the combination of data from source and target domains to train the model and (2) KLD-based model adaptation (referred to as `kld`), where we first trained a model on the source data and later adapted it to the target domain with KLD regularization. In both cases, CHiME4 was used as the source dataset and VoCub as the target dataset.

The architecture and training procedure of both models follow the baseline ones, except for following modifications. For the `multi-condition` strategy, we trained the model on the combination of CHiME4 and VoCub datasets. For the `kld` strategy, we first trained the network on CHiME4, then adapted it to VoCub dataset. For the adaptation phase, we added KLD regularization in the loss. The weight λ applied to the KLD regularization term was tuned on the validation set considering only the FER, and the other optimization parameters are reset to their original values. The output dimension of the two models is 1808 and 1984 respectively.

The performance of the two strategies on the validation set is shown in Table 6.1, together with the baseline system. As a comparison, I also present results obtained with a state of the art commercial product, Google Cloud Speech-to-Text[7] (referred to as `google`). Google product can also take hints as input, in the form of a list of sentences which are likely to appear in the evaluation set of utterances. This biases the model toward the specified sentences (without excluding completely sentences that are not in the list) and should provide better results. Scores with our list of commands as hints is also shown (referred to as `google-hinted`).

It can be seen from the table that the baseline model, despite being trained on a small amount of data, is hard to beat. The gap is particularly important with Google model, which gives poor results with 75.2% and 62.5% of SER respectively without and with hints. This low performance can easily be explained as the model was trained without domain-matched data and uses an unconstrained grammar (or a biased one when hints are provided). The comparison is quite unfair in this respect, but illustrates the difficulty of training a model that

[7] Available at https://cloud.google.com/speech-to-text/

TABLE 6.1 Error rate (%) of the different models on VoCub validation set.

Model	**WER (%)**	**SER (%)**
`google`	50.72	75.21
`google-hinted`	39.43	62.50
baseline	**5.62**	**13.56**
`multi-condition`	7.48	16.03
`kld`	9.69	18.94

TABLE 6.2 Error rate (%) of the baseline and the two adaptation strategies with a word loop grammar on VoCub validation set.

Model	**WER (%)**	**SER (%)**
baseline	28.64	63.35
`multi-condition`	**28.24**	**57.84**
`kld`	28.57	65.53

is robust over a large range of speakers and acoustic conditions. Even commercial systems trained on very large datasets struggle with peculiar conditions such as the one considered here. This further motivates the approach taken in this thesis.

Surprisingly, both adaptation strategies, multi-condition training and KLD-based model adaptation, are outperformed by the baseline in this condition. VoCub training set seems big enough to obtain reliable performance when used alone.

Table 6.2 show more nuanced results using the loop grammar already described in section 3.2.5 (which allows any combination of the 62 words from the dataset). As can be seen, using a weaker grammar (giving a better evaluation of the acoustic model) multi-condition training performs better than the baseline, providing an improvement of 9% relative on the SER. The `kld` strategy still performs worse than the baseline but by a smaller margin (about 3% relative) when evaluated this way. This is encouraging and shows that multi-condition training has a positive effect on the performance of the acoustic model, despite its performance being worse when used in conjunction with the strict grammar.

Strangely, this differences in SER are obtained despite very similar WER for the three models. In order to explain this phenomenon, in Table 6.3, I analyze separately the three types of errors composing the WER: insertions, deletions and substitutions. We can observe that the acoustic models have very different patterns of error. The `multi-condition` strategy has the highest number of substitutions but the lowest number of insertions and deletions. The baseline and the `kld` strategy result in the highest number of insertions and deletions

TABLE 6.3 Decomposition of the WER on VoCub validation set with a loop grammar into insertions, deletions and substitutions. Results are presented for the baseline and the two adaptation strategies.

Model	**Insertions**	**Deletions**	**Substitutions**
baseline	301	73	263
`multi-condition`	**229**	**54**	345
`kld`	245	123	**260**

TABLE 6.4 Error rate (%) of the baseline and the `multi-condition` strategy on VoCub test set.

Model	**WER (%)**	**SER (%)**
baseline	4.14	7.64
`multi-condition`	**2.52**	**5.51**

respectively, while showing very close number of substitutions. These differences are likely to explain the variations in SER.

Test results of the baseline and the best adaptation strategy, shown in Table 6.4, further comfort the use of the `multi-condition` strategy. Unlike what is observed for the validation test, multi-condition training here outperforms the baseline by 28% relative.

Finally, while the use of domain adaptation was mainly aiming at improving the accuracy on the core VoCub commands, it can also be expected that the model will generalize better to new ones. It has indeed been shown in section 3.2.5 that the performance of the baseline system deteriorates when tested on sentences not seen at training time (conditions 3 and 4 of the dataset). Table 6.5 shows again the performance of the baseline on the two groups, together with the performance of the `multi-condition` strategy in the same context. The extended grammar (defined in section A.2) is used in this case.

TABLE 6.5 Evaluation of the baseline and the multi-condition training strategy on VoCub extended test set. WER and SER are given for conditions 1 and 2 on one side and 3 and 4 on the other side.

Model	**Conditions 1 and 2**		**Conditions 3 and 4**	
	WER (%)	**SER (%)**	**WER (%)**	**SER (%)**
baseline	**3.47**	7.22	66.08	88.05
`multi-condition`	3.91	**6.57**	**54.17**	**80.92**

As can be seen from the table, the `multi-condition` strategy achieves better SER for both groups of sentences. On the new sentences, the adaptation strategy performs 8% relatively better than the baseline, showing better generalization capabilities.

Overall, multi-condition training proves to be a useful strategy to improve the performance of the acoustic model on the original sentences (despite giving worse performance on the validation set with the strict grammar) and generalizes better to new sentences. It can be expected also that it will be more robust to background noise (not evaluated here).

6.4 Demonstration on the robot

6.4.1 Application to the IOL demo with iCub

The first version of iCubrec was tested with the IOL demo on iCub. Audio was acquired through the same tablet that was used to acquire VoCub dataset. The VAD was not present at that time and a manual trigger was used to signal starting and ending point of each command. A video illustrating the obtained result can be found at https://youtu.be/xdzdQYGwBFg.

6.4.2 Application to the online detection demo with R1

The second version of the pipeline has been tested with the online detection demo[8] on R1. This demo uses a slightly reduced grammar (see section A.3). While the tablet is still used to acquire the audio, the VAD system provides a completely hand-free experience. As for the acoustic model, the DNN obtained through the `multi-condition` strategy presented earlier has been retained. A video illustrating the obtained result can be found at https://youtu.be/kvyVaLfILbY.

6.5 Future work

In this chapter, I presented the iCubrec command recognition system we developed. It allows reliable and hand-free vocal interaction with the robots, which was one of the main goals of my PhD. Using multi-condition training, the performance of the acoustic system where improved in two ways. First, despite worse performance on the validation set with the strict grammar, overall performance of the acoustic model on VoCub core command is improved. Also, the model generalizes better to new commands.

[8]The online detection demo improves on the IOL demo by using a DNN-based visual detection system (instead of a simple heuristic) and incorporates a stronger classification system.

From there, several improvements are possible:

- The system was only tested using a tablet for audio signal acquisition, the same tablet that was used to record VoCub dataset. While replacing the tablet microphone with one of the robot microphones is straightforward technically, we may observe a deterioration of performance due to different channel characteristics. This will have to be evaluated.

- This brings me to a limit of the evaluation performed in subsection 6.2.1 and section 6.3, which only considered the VAD and the acoustic model in isolation. The interaction of the two modules are not tested, nor is the performance of the system in continuous streaming conditions with silent periods and non-command speech. The design and implementation of an evaluation protocol for the full pipeline (including the filler and rejection mechanisms, as well as the possible use of a different microphone) would allow to assess better the performance of the system. In that case, evaluation metrics from KWS literature will probably be more appropriate.

- Regarding domain adaptation, we only explored multi-condition training and KLD-based model adaptation here. More advanced techniques, such as the ones analyzed in chapter 4 would probably lead to further improvement. The robustness of the system is key in the user's experience. It would thus be very desirable to push the accuracy of the system further. With 5.5% of mis-recognized commands, there is still a lot of room for progress in the acoustic model.

- When spontaneously interacting with the robot, users are very likely to introduce disfluencies or use variants of the commands we specified. It would be very interesting to explore the robustness of the system to these sources of perturbation. While it is possible to improve the acceptance rate of those sentences (which depends mainly on the rejection threshold and the entry cost of the garbage model) this will also likely increase false positives. Alternatively, natural language processing techniques could be used to relax the one-to-one constraint between vocal commands and meaning, allowing more flexibility in the expression of the user's intent.

- Finally, it has been shown that performance can deteriorate significantly when new sentences, not present in the training set, are added to the system. This strongly limits the reusability of the system for different applications. The work on sequence-to-sequence models described in chapter 5 is an attempt to solve this problem, by providing a simpler training and deployment procedure. This would allow non-expert users to gather their own dataset and train a dedicated system on it. The few-shot

strategies I proposed also explore the possibility to extend an existing system with a minimal amount of data for the new commands. It would be interesting to evaluate how these approaches perform in real conditions.

Chapter 7

Conclusions and future work

7.1 Contributions

In this thesis, I explored the use of spoken commands to control a robot. While robustness was my first concern, a particular attention has been put on building a simple pipeline that can be reused more easily by the robotics community.

In order to reach high accuracy despite scarce resource conditions, domain adaptation strategies were investigated. Two methods based on the O-MTL paradigm (Badino et al., 2017) have been proposed. The first one considered the joint optimization of a gender-independent network with two gender-dependent models. Experiments on TIMIT showed mitigated results. While the method proved useful with 3.2% of improvement on the FER, this only partially carried over to the PER. A second approach has been proposed where the target network was jointly optimized with a model having access to speaker information. This second approach also showed limited results on the PER with only 0.3% of improvement. Better optimization strategies may be useful to get the most out of this paradigm. It would also be interesting to test the same approaches on a more challenging task such as CHiME4 dataset, which also includes noisy conditions.

In order to simplify the command recognition pipeline, a second contribution explored the use of the attention-based encoder-decoder architecture. The S2S model showed very good results, improving keyword recognition by 40% over a strong CNN-HMM baseline. I additionally explored the possibility to introduce new keywords with only few examples per keyword (few-shot learning strategies). A first strategy that requires to retrain the model from scratch achieved 40.5% of recognition on the new keywords with only 10 examples per keyword and 81.5% when 100 examples are available. A faster alternative based on model adaptation yielded even better results on the new keywords with 68.8% and 88.4% of

accuracy for 10 and 100 keywords respectively. The latter results are achieved at the expense of the performance on the original categories though. These results are very promising as they open the possibility to easily include new keywords in the command recognition system.

Finally, I described the iCubrec pipeline which was developed to test spoken command recognition on a real scenario. For this purpose, the VoCub dataset has been gathered corresponding to $\sim$2.5 hours of data matching the test scenario. Using this dataset, DNN-based acoustic models have been trained. The best performing model, trained under the multi-condition training paradigm, achieved 94.5% of accuracy on the test set. The fully functional pipeline, which includes a VAD system for a completely hand-free interaction, has been successfully tested on R1 robot.

7.2 Future work

Unfortunately, none of the approaches studied in this thesis for domain adaptation of few-shot learning with S2S models found there way in iCubrec pipeline. Despite being very effective, multi-condition training remains a simple domain adaptation strategy. It would be interesting, in the near future, to explore more advanced strategies to further improve the accuracy of iCubrec system, which is key in the user's experience. It would also be useful to establish an evaluation protocol for the full pipeline, including the VAD module, to get a more accurate estimation of its performance in real usage conditions.

While the attention-based encoder-decoder architecture showed promising results for keyword classification, it's application to more complex commands should be further investigated. The possibility to use S2S models in place of the more complex DNN-HMM system, opens appealing perspectives in terms of reusability. The adaptation of this paradigm to online application, through techniques such as windowed attention or online-enabled BiLSTM units, also seems promising.

Finally, the possibility to extend the vocabulary with minimal data requirements – e.g. through few-shot strategies – is another attractive direction for future work. Going one step further, it would be interesting to investigate incremental learning strategies, such as the one proposed by Camoriano et al. (2017) for vision, where new commands could progressively be learned through interaction with the robot. One of the main challenge then is to acquire labels for the new audio examples. Camoriano et al. assumed an existing speech recognition system to provide labels for the new objects the system has to learn visually. Conversely, audio labels could be acquired relying on another modality. This raises an egg-and-chicken problem though if the different modalities are relying on one another.

7.3 Perspectives

While smartphone or tablets usually offer other type of interfaces (such as tactile interfaces), vocal interactions are going to be indispensable for many applications in robotics. On the other hand, robotic platforms offer many promising features which could be exploited to improve command recognition. The most obvious modality that can be leveraged is vision. Robots are readily equipped with cameras, allowing synergies between auditive and visual information. Vision could for example be used to provide side information about, e.g, the speaker. Visual information could also be more fully integrated in the auditive pipeline to improve VAD or command recognition. Hence, Savran et al. (2018) showed how visual detection of lip activity can improve VAD. Similarly, Ephrat et al. (2018) or Morrone et al. (2018) explored the use of vision for speech enhancement in multi-talker environments.

Another advantage of robotic platforms is their movement capabilities, which are quite unique compared to other speech-enabled devices. For example, work on the iCub by Hambrook et al. (2017) showed how head movements can help localize sounds. This can in turn be used to filter sound from other directions improving the quality of the audio signal for latter processing. Another big challenge in command recognition is to determine whether speech is addressed to the vocal interface or not. This is an ill-posed problem when considering only audio information. A robotic platform could leverage on visual information to detect whether the speaker is rather speaking to the robot or addressing another human. In this context head movements could help keep the speaker in the robot's field of view.

To conclude, as vocal interfaces will continue to improve, their understanding of the context of a command will become more crucial to respond appropriately. The capability of the system to access other modalities (especially vision) and maybe to act on the environment will become more important. Moving from current deep learning paradigm that performs very well on isolated tasks (such as object detection, speech recognition or machine translation) to a more holistic apprehension of the world is probably the next challenge for machine learning. In this context, robotic platforms are an ideal tool, enabling the study of the different components in interaction.

References

Abadi, Martín, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. (2016). "Tensorflow: Large-scale machine learning on heterogeneous distributed systems". In: *arXiv preprint arXiv:1603.04467* (see p. 34).

Abdel-Hamid, Ossama, Li Deng, and Dong Yu (2013). "Exploring convolutional neural network structures and optimization techniques for speech recognition." In: *Interspeech*. Vol. 2013, pp. 1173–5 (see p. 17).

Abrash, Victor, Horacio Franco, Ananth Sankar, and Michael Cohen (1995). "Connectionist speaker normalization and adaptation". In: *Fourth European Conference on Speech Communication and Technology* (see p. 51).

Anastasakos, Tasos, John McDonough, Richard Schwartz, and John Makhoul (1996). "A compact model for speaker-adaptive training". In: *Spoken Language, 1996. ICSLP 96. Proceedings., Fourth International Conference on*. Vol. 2. IEEE, pp. 1137–1140 (see p. 51).

Asami, T., R. Masumura, Y. Yamaguchi, H. Masataki, and Y. Aono (2017). "Domain adaptation of DNN acoustic models using knowledge distillation". In: *Proc. of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 5185–5189. DOI: 10.1109/ICASSP.2017.7953145 (see p. 53).

Audhkhasi, Kartik, Brian Kingsbury, Bhuvana Ramabhadran, George Saon, and Michael Picheny (2017). "Building competitive direct acoustics-to-word models for English conversational speech recognition". In: *arXiv:1712.03133 [cs, stat]*. arXiv: 1712.03133 (see pp. 25, 29).

Audhkhasi, Kartik, Bhuvana Ramabhadran, George Saon, Michael Picheny, and David Nahamoo (2017). "Direct Acoustics-to-Word Models for English Conversational Speech Recognition". In: *arXiv:1703.07754 [cs, stat]*. arXiv: 1703.07754 (see pp. 25, 29).

Badino, Leonardo, Luca Franceschi, Raman Arora, Michele Donini, and Massimiliano Pontil (2017). "A Speaker Adaptive DNN Training Approach for Speaker-Independent Acoustic Inversion". In: *Proc. of Interspeech*. Stockholm, Sweden, pp. 984–988 (see pp. 54–57, 59, 111).

Bahdanau, D., J. Chorowski, D. Serdyuk, P. Brakel, and Y. Bengio (2016). "End-to-end attention-based large vocabulary speech recognition". In: *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4945–4949. DOI: 10.1109/ICASSP.2016.7472618 (see pp. 24, 25).

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio (2015). "Neural Machine Translation by Jointly Learning to Align and Translate". In: *Proc. of the International Conference on Learning Representations (ICLR)*. arXiv: 1409.0473. San Diego, CA, USA (see p. 23).

Bahl, L., P. Brown, P. de Souza, and R. Mercer (1986). "Maximum mutual information estimation of hidden Markov model parameters for speech recognition". In: *ICASSP '86.*

IEEE International Conference on Acoustics, Speech, and Signal Processing. Vol. 11, pp. 49–52. DOI: 10.1109/ICASSP.1986.1169179 (see p. 18).

Baum, Leonard E. and Ted Petrie (1966). "Statistical Inference for Probabilistic Functions of Finite State Markov Chains". In: *The Annals of Mathematical Statistics* 37.6, pp. 1554–1563. ISSN: 0003-4851 (see p. 13).

Bengio, Y., P. Simard, and P. Frasconi (1994). "Learning long-term dependencies with gradient descent is difficult". In: *IEEE Transactions on Neural Networks* 5.2, pp. 157–166. ISSN: 1045-9227. DOI: 10.1109/72.279181 (see p. 17).

Bottou, Léon (1998). "Online learning and stochastic approximations". In: *On-line learning in neural networks* 17.9, p. 142 (see p. 20).

Bourlard, H., B. D'hoore, and J.-M. Boite (1994). "Optimizing recognition and rejection performance in wordspotting systems". In: *Proc. of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. Vol. 1, pp. 373–376. DOI: 10.1109/ICASSP.1994.389278 (see pp. 29–31).

Bourlard, H. and C. J. Wellekens (1990). "Links between Markov models and multilayer perceptrons". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 12.12, pp. 1167–1178. ISSN: 0162-8828. DOI: 10.1109/34.62605 (see p. 15).

Bucilă, Cristian, Rich Caruana, and Alexandru Niculescu-Mizil (2006). "Model Compression". In: *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '06. New York, NY, USA: ACM, pp. 535–541. ISBN: 1-59593-339-5. DOI: 10.1145/1150402.1150464 (see p. 52).

Camoriano, R., G. Pasquale, C. Ciliberto, L. Natale, L. Rosasco, and G. Metta (2017). "Incremental robot learning of new objects with fixed update time". In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3207–3214. DOI: 10.1109/ICRA.2017.7989364 (see p. 112).

Chan, W., N. Jaitly, Q. Le, and O. Vinyals (2016). "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition". In: *Proc. of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. Shanghai, China, pp. 4960–4964. DOI: 10.1109/ICASSP.2016.7472621 (see pp. 23, 24, 78).

Chan, William, Yu Zhang, Quoc Le, and Navdeep Jaitly (2017). "Latent Sequence Decompositions". In: *Proc. of the International Conference on Learning Representations (ICLR)*. arXiv: 1610.03035 (see p. 25).

Chen, G., C. Parada, and G. Heigold (2014). "Small-footprint keyword spotting using deep neural networks". In: *Proc. of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 4087–4091. DOI: 10.1109/ICASSP.2014.6854370 (see pp. 8, 30, 75).

Chiu, C., T. N. Sainath, Y. Wu, R. Prabhavalkar, P. Nguyen, Z. Chen, A. Kannan, R. J. Weiss, K. Rao, E. Gonina, N. Jaitly, B. Li, J. Chorowski, and M. Bacchiani (2018). "State-of-the-Art Speech Recognition with Sequence-to-Sequence Models". In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4774–4778. DOI: 10.1109/ICASSP.2018.8462105 (see pp. 23, 25).

Chorowski, Jan, Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio (2014). "End-to-end Continuous Speech Recognition using Attention-based Recurrent NN: First Results". In: *presented at NIPS Workshop on Deep Learning and Representation Learning*. arXiv: 1412.1602. Montréal, Canada (see pp. 23, 24).

Chorowski, Jan, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio (2015). "Attention-based models for speech recognition". In: *Advances in Neural Information Processing Systems (NIPS)*. Montréal, Canada, pp. 577–585 (see pp. 24, 78).

Chorowski, Jan and Navdeep Jaitly (2016). "Towards better decoding and language model integration in sequence to sequence models". In: *arXiv:1612.02695 [cs, stat]*. arXiv: 1612.02695 (see p. 25).

Ciliberto, Carlo (2017). "Connecting YARP to the Web with Yarp.js". English. In: *Frontiers in Robotics and AI* 4. ISSN: 2296-9144. DOI: 10.3389/frobt.2017.00067 (see p. 101).

Cui, Xiaodong, Vaibhava Goel, and George Saon (2017). "Embedding-Based Speaker Adaptive Training of Deep Neural Networks". In: (see p. 49).

Damerau, F (1971). *Markov models and linguistic theory: an experimental study of a model for English*. Mouton. Janua Linguarum. Series Minor (see p. 16).

Davis, S. and P. Mermelstein (1980). "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences". In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 28.4, pp. 357–366. ISSN: 0096-3518. DOI: 10.1109/TASSP.1980.1163420 (see p. 11).

Dehak, Najim, Patrick J Kenny, Réda Dehak, Pierre Dumouchel, and Pierre Ouellet (2011). "Front-end factor analysis for speaker verification". In: *IEEE Transactions on Audio, Speech, and Language Processing* 19.4, pp. 788–798 (see p. 49).

Dempster, A. P., N. M. Laird, and D. B. Rubin (1977). "Maximum Likelihood from Incomplete Data via the EM Algorithm". In: *Journal of the Royal Statistical Society. Series B (Methodological)* 39.1, pp. 1–38. ISSN: 0035-9246 (see p. 13).

Deng, L., K. Hassanein, and M. Elmasry (1994). "Analysis of the correlation structure for a neural predictive model with application to speech recognition". In: *Neural Networks* 7.2, pp. 331–339. ISSN: 0893-6080. DOI: 10.1016/0893-6080(94)90027-2 (see p. 17).

Duchi, John, Elad Hazan, and Yoram Singer (2011). "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization". In: *Journal of Machine Learning Research* 12.Jul, pp. 2121–2159. ISSN: ISSN 1533-7928 (see p. 21).

Eide, E. and H. Gish (1996). "A parametric approach to vocal tract length normalization". In: *Proc. of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. Vol. 1, 346–348 vol. 1. DOI: 10.1109/ICASSP.1996.541103 (see p. 50).

Ephrat, Ariel, Inbar Mosseri, Oran Lang, Tali Dekel, Kevin Wilson, Avinatan Hassidim, William T. Freeman, and Michael Rubinstein (2018). "Looking to Listen at the Cocktail Party: A Speaker-Independent Audio-Visual Model for Speech Separation". en. In: *Proc. of SIGGRAPH*. DOI: 10.1145/3197517.3201357 (see p. 113).

Evgeniou, Theodoros and Massimiliano Pontil (2004). "Regularized Multi–task Learning". In: *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '04. Seattle, WA, USA: ACM, pp. 109–117. ISBN: 1-58113-888-1. DOI: 10.1145/1014052.1014067 (see pp. 54, 55).

Falavigna, Daniele, Marco Matassoni, Shahab Jalalvand, Matteo Negri, and Marco Turchi (2017). "DNN adaptation by automatic quality estimation of ASR hypotheses". In: *Computer Speech & Language* 46, pp. 585–604. ISSN: 0885-2308. DOI: 10.1016/j.csl.2016.11.002 (see p. 51).

Fukuda, Takashi, Masayuki Suzuki, Gakuto Kurata, Samuel Thomas, Jia Cui, and Bhuvana Ramabhadran (2017). "Efficient Knowledge Distillation from an Ensemble of Teachers". In: *Proc. of Interspeech*. Stockholm, Sweden, pp. 3697–3701 (see p. 53).

Gales, Mark JF et al. (1998). "Maximum likelihood linear transformations for HMM-based speech recognition". In: *Computer speech & language* 12.2, pp. 75–98 (see p. 50).

Garcia, A. and Herbert Gish (2006). "Keyword Spotting of Arbitrary Words Using Minimal Speech Resources". In: *Proc. of the IEEE International Conference on Acoustics, Speech,*

and Signal Processing (ICASSP). Vol. 1, pp. I–I. DOI: 10.1109/ICASSP.2006.1660179 (see p. 27).

Garofolo, John S, Lori F Lamel, William M Fisher, Jonathan G Fiscus, David S Pallett, Nancy L Dahlgren, and Victor Zue (1993). "TIMIT Acoustic-Phonetic Continuous Speech Corpus LDC93S1". In: *Web Download. Philadelphia: Linguistic Data Consortium* (see pp. 36, 37).

Garofolo, John, D Graff, D Paul, and D Pallett (1993). "CSR-I (WSJ0) Complete LDC93S6A". In: *Web Download. Philadelphia: Linguistic Data Consortium* (see pp. 36, 37).

Gemello, Roberto, Franco Mana, Stefano Scanzio, Pietro Laface, and Renato De Mori (2007). "Linear hidden transformations for adaptation of hybrid ANN/HMM models". In: *Speech Communication* 49.10-11, pp. 827–835 (see pp. 50, 51).

Ghoshal, A., P. Swietojanski, and S. Renals (2013). "Multilingual training of deep neural networks". In: *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 7319–7323. DOI: 10.1109/ICASSP.2013.6639084 (see p. 54).

Gibson, Matthew and Thomas Hain (2006). "Hypothesis spaces for minimum bayes risk training in large vocabulary speech recognition". In: *Proc. of the Ninth International Conference on Spoken Language Processing*. Pittsburgh, PA, USA (see pp. 18, 81).

Graves, A., N. Jaitly, and A. Mohamed (2013). "Hybrid speech recognition with Deep Bidirectional LSTM". In: *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*, pp. 273–278. DOI: 10.1109/ASRU.2013.6707742 (see p. 17).

Graves, Alex (2012). "Sequence Transduction with Recurrent Neural Networks". In: *presented at the International Conference of Machine Learning (ICML) Workshop on Representation Learning*. arXiv: 1211.3711. Edinburgh, Scotland (see p. 23).

Graves, Alex, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber (2006). "Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks". In: *Proc. of the 23rd International Conference on Machine Learning*. ICML'06. New York, NY, USA: ACM, pp. 369–376. ISBN: 978-1-59593-383-6. DOI: 10.1145/1143844.1143891 (see pp. 19, 22).

Graves, Alex and Navdeep Jaitly (2014). "Towards End-to-end Speech Recognition with Recurrent Neural Networks". In: *Proc. of the 31st International Conference on Machine Learning*. Vol. 32. ICML'14. Beijing, China, pp. II–1764–II–1772 (see p. 23).

Hambrook, Dillon A., Marko Ilievski, Mohamad Mosadeghzad, and Matthew Tata (2017). "A Bayesian computational basis for auditory selective attention using head rotation and the interaural time-difference cue". en. In: *PLOS ONE* 12.10, e0186104. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0186104 (see p. 113).

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun (2016). "Deep Residual Learning for Image Recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. Seattle, WA, USA: IEEE, pp. 770–778. ISBN: 978-1-4673-8851-1. DOI: 10.1109/CVPR.2016.90 (see p. 17).

He, Y., R. Prabhavalkar, K. Rao, W. Li, A. Bakhtin, and I. McGraw (2017). "Streaming Small-Footprint Keyword Spotting using Sequence-to-Sequence Models". In: *ArXiv e-prints* (see pp. 28, 74).

Heigold, G., V. Vanhoucke, A. Senior, P. Nguyen, M. Ranzato, M. Devin, and J. Dean (2013). "Multilingual acoustic models using distributed deep neural networks". In: *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 8619–8623. DOI: 10.1109/ICASSP.2013.6639348 (see p. 54).

Hermansky, Hynek, Daniel PW Ellis, and Sangita Sharma (2000). "Tandem connectionist feature extraction for conventional HMM systems". In: *icassp*. IEEE, pp. 1635–1638 (see p. 15).

Higy, Bertrand and Peter Bell (2018). "Few-shot learning with attention-based sequence-to-sequence models". In: arXiv: 1811.03519 (see pp. x, 73).

Higy, Bertrand, Alessio Mereta, Giorgio Metta, and Leonardo Badino (2018). "Speech Recognition for the iCub Platform". In: *Frontiers in Robotics and AI* 5. ISSN: 2296-9144. DOI: 10.3389/frobt.2018.00010 (see pp. x, 36, 40, 94, 103).

Hinton, G., Li Deng, Dong Yu, G.E. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T.N. Sainath, and B. Kingsbury (2012). "Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups". In: *IEEE Signal Processing Magazine* 29.6, pp. 82–97. ISSN: 1053-5888. DOI: 10.1109/MSP.2012.2205597 (see p. 15).

Hinton, Geoffrey, Oriol Vinyals, and Jeff Dean (2015). "Distilling the Knowledge in a Neural Network". In: *arXiv:1503.02531 [cs, stat]*. arXiv: 1503.02531 (see pp. 52, 53, 58).

Hochreiter, Sepp and Jürgen Schmidhuber (1997). "Long Short-Term Memory". In: *Neural Computation* 9.8, pp. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735 (see p. 17).

Hori, Takaaki, Shinji Watanabe, Yu Zhang, and William Chan (2017). "Advances in Joint CTC-Attention based End-to-End Speech Recognition with a Deep CNN Encoder and RNN-LM". In: *Proc. of Interspeech*. arXiv: 1706.02737. Stockholm, Sweden (see pp. 24, 36, 78, 80).

Huang, J., J. Li, D. Yu, L. Deng, and Y. Gong (2013). "Cross-language knowledge transfer using multilingual deep neural network with shared hidden layers". In: *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 7304–7308. DOI: 10.1109/ICASSP.2013.6639081 (see p. 54).

Huang, Xuedong, Alex Acero, and Hsiao-Wuen Hon (2001). *Spoken Language Processing: A Guide to Theory, Algorithm and System Development*. English. 1 edition. Upper Saddle River, NJ: Prentice Hall. ISBN: 978-0-13-022616-7 (see p. 10).

Jaitly, Navdeep, Quoc V Le, Oriol Vinyals, Ilya Sutskever, David Sussillo, and Samy Bengio (2016). "An Online Sequence-to-Sequence Model Using Partial Conditioning". In: *Advances in Neural Information Processing Systems (NIPS)*. Ed. by D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett. Curran Associates, Inc., pp. 5067–5075 (see p. 25).

Joy, Neethu Mariam, Sandeep Reddy Kothinti, S Umesh, and Basil Abraham (2017). "Generalized distillation framework for speaker normalization". In: *Proc. of Interspeech*. Stockholm, Sweden (see p. 53).

Junkawitsch, Jochen, L. Neubauer, Harald Hoge, and Günther Ruske (1996). "A new keyword spotting algorithm with pre-calculated optimal thresholds". In: *Proc. of the Fourth International Conference on Spoken Language, 1996. ICSLP 96. Proceedings*. Vol. 4, 2067–2070 vol.4. DOI: 10.1109/ICSLP.1996.607208 (see p. 31).

Kaiser, Janez, Bogomir Horvat, and Zdravko Kacic (2000). "A novel loss function for the overall risk criterion based discriminative training of HMM models". In: *Sixth International Conference on Spoken Language Processing* (see p. 18).

Keshet, Joseph, David Grangier, and Samy Bengio (2009). "Discriminative keyword spotting". en. In: *Speech Communication* 51.4, pp. 317–329. ISSN: 01676393. DOI: 10.1016/j.specom.2008.10.002 (see p. 31).

Khosla, Aditya, Tinghui Zhou, Tomasz Malisiewicz, Alexei Efros, and Antonio Torralba (2012). "Undoing the Damage of Dataset Bias". In: *Proceedings of the 12th European Conference on Computer Vision* (see pp. 54, 55).

Kim, S., Takaaki Hori, and Shinji Watanabe (2017). "Joint CTC-attention based end-to-end speech recognition using multi-task learning". In: *Proc. of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. New Orleans, LA, USA, pp. 4835–4839. DOI: 10.1109/ICASSP.2017.7953075 (see pp. 25, 36, 54, 76, 80).

Kingma, Diederik P. and Jimmy Ba (2015). "Adam: A Method for Stochastic Optimization". In: *Proc. of the 3rd International Conference for Learning Representations*. arXiv: 1412.6980. San Diego, CA, USA (see pp. 22, 102).

Lecun, Yann, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel (1990). "Handwritten digit recognition with a back-propagation network". English (US). In: *Advances in Neural Information Processing Systems (NIPS),* Denver, CO (see p. 17).

Lee, K.- and H.- Hon (1989). "Speaker-independent phone recognition using hidden Markov models". In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 37.11, pp. 1641–1648. ISSN: 0096-3518. DOI: 10.1109/29.46546 (see p. 58).

Li, Bo and Khe Chai Sim (2010). "Comparison of discriminative input and output transformations for speaker adaptation in the hybrid NN/HMM systems". In: *Eleventh Annual Conference of the International Speech Communication Association* (see p. 51).

Li, J., D. Yu, J. Huang, and Y. Gong (2012). "Improving wideband speech recognition using mixed-bandwidth training data in CD-DNN-HMM". In: *2012 IEEE Spoken Language Technology Workshop (SLT)*, pp. 131–136. DOI: 10.1109/SLT.2012.6424210 (see p. 9).

Li, Jinyu, Michael L Seltzer, Xi Wang, Rui Zhao, and Yifan Gong (2017). "Large-Scale Domain Adaptation via Teacher-Student Learning". In: *Proc. of Interspeech*. Stockholm, Sweden, pp. 2386–2390 (see p. 53).

Li, Jinyu, Guoli Ye, Amit Das, Rui Zhao, and Yifan Gong (2018). "Advancing Acoustic-to-Word CTC Model". In: *Proc. of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. arXiv: 1803.05566 (see pp. 25, 29).

Li, Jinyu, Rui Zhao, Jui-Ting Huang, and Yifan Gong (2014). "Learning small-size DNN with output-distribution-based criteria". In: *Fifteenth annual conference of the international speech communication association* (see p. 52).

Liao, H. (2013). "Speaker adaptation of context dependent deep neural networks". In: *Proc. of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 7947–7951. DOI: 10.1109/ICASSP.2013.6639212 (see pp. 48, 50).

Lopez-Paz, David, Léon Bottou, Bernhard Schölkopf, and Vladimir Vapnik (2015). "Unifying distillation and privileged information". In: *arXiv:1511.03643 [cs, stat]*. arXiv: 1511.03643 (see p. 53).

Manos, A.S. and V.W. Zue (1997). "A segment-based wordspotter using phonetic filler models". In: *Proc. of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. Vol. 2, 899–902 vol.2. DOI: 10.1109/ICASSP.1997.596081 (see p. 29).

Markov, Konstantin and Tomoko Matsui (2016). "Robust Speech Recognition Using Generalized Distillation Framework." In: *Proc. of Interspeech*. San Francisco, CA, USA (see p. 53).

Metta, Giorgio, Paul Fitzpatrick, and Lorenzo Natale (2006). "YARP: yet another robot platform". In: *International Journal of Advanced Robotic Systems* 3.1, p. 8 (see p. 95).

Metta, Giorgio, Lorenzo Natale, Francesco Nori, Giulio Sandini, David Vernon, Luciano Fadiga, Claes von Hofsten, Kerstin Rosander, Manuel Lopes, José Santos-Victor, Alexandre Bernardino, and Luis Montesano (2010). "The iCub humanoid robot: An open-systems platform for research in cognitive development". In: *Neural Networks*. Social Cognition: From Babies to Robots 23.8, pp. 1125–1134. ISSN: 0893-6080. DOI: 10.1016/j.neunet.2010.08.010 (see pp. 6, 94).

Mikolov, Tomáš, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur (2010). "Recurrent neural network based language model". In: *Eleventh Annual Conference of the International Speech Communication Association* (see p. 16).

Miller, David R. H., Michael Kleber, Kao Chia-lin, Owen Kimball, Thomas Colthurst, Stephen A. Lowe, Richard M. Schwartz, and Herbert Gish (2007). "Rapid and accurate spoken term detection". In: *Eighth Annual Conference of the International Speech Communication Association* (see p. 27).

Mimura, M., S. Sakai, and T. Kawahara (2017). "Cross-domain speech recognition using nonparallel corpora with cycle-consistent adversarial networks". In: *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pp. 134–140. DOI: 10.1109/ASRU.2017.8268927 (see p. 50).

Mohamed, A., G. Hinton, and G. Penn (2012). "Understanding how Deep Belief Networks perform acoustic modelling". In: *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4273–4276. DOI: 10.1109/ICASSP.2012.6288863 (see p. 9).

Morrone, Giovanni, Luca Pasa, Vadim Tikhanoff, Sonia Bergamaschi, Luciano Fadiga, and Leonardo Badino (2018). "Face Landmark-based Speaker-Independent Audio-Visual Speech Enhancement in Multi-Talker Environments". In: *arXiv:1811.02480 [cs, eess]*. arXiv: 1811.02480 (see p. 113).

Nair, Vinod and Geoffrey E Hinton (2010). "Rectified linear units improve restricted boltzmann machines". In: *Proceedings of the 27th international conference on machine learning (ICML-10)*, pp. 807–814 (see p. 17).

Neto, Joao, Luís Almeida, Mike Hochberg, Ciro Martins, Luis Nunes, Steve Renals, and Tony Robinson (1995). "Speaker-adaptation for hybrid HMM-ANN continuous speech recognition system". In: *Fourth European Conference on Speech Communication and Technology* (see p. 51).

Ochiai, T., S. Matsuda, X. Lu, C. Hori, and S. Katagiri (2014). "Speaker Adaptive Training using Deep Neural Networks". In: *Proc. of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 6349–6353. DOI: 10.1109/ICASSP.2014.6854826 (see p. 51).

Ochiai, T., S. Matsuda, H. Watanabe, X. Lu, C. Hori, and S. Katagiri (2015). "Speaker adaptive training for deep neural networks embedding linear transformation networks". In: *Proc. of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 4605–4609. DOI: 10.1109/ICASSP.2015.7178843 (see p. 51).

Palaz, Dimitri, Ronan Collobert, and Mathew Magimai -Doss (2013). "Estimating Phoneme Class Conditional Probabilities from Raw Speech Signal using Convolutional Neural Networks". In: *Proc. of Interspeech*. arXiv: 1304.1018 (see p. 9).

Povey, D. and B. Kingsbury (2007). "Evaluation of Proposed Modifications to MPE for Large Scale Discriminative Training". In: *2007 IEEE International Conference on Acoustics, Speech and Signal Processing - ICASSP '07*. Vol. 4, pp. IV–321–IV–324. DOI: 10.1109/ICASSP.2007.366914 (see p. 18).

Povey, Daniel (2004). "Discriminative training for large vocabulary speech recognition". PhD Thesis. University of Cambridge (see p. 18).

Povey, Daniel, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, et al. (2011). "The Kaldi speech recognition toolkit". In: *IEEE 2011 workshop on automatic speech recognition and understanding*. Waikoloa Village, HI, US: IEEE Signal Processing Society (see p. 34).

Povey, Daniel, Vijayaditya Peddinti, Daniel Galvez, Pegah Ghahremani, Vimal Manohar, Xingyu Na, Yiming Wang, and Sanjeev Khudanpur (2016). "Purely Sequence-Trained Neural Networks for ASR Based on Lattice-Free MMI." In: *Interspeech*, pp. 2751–2755 (see p. 19).

Prabhavalkar, R., R. Alvarez, C. Parada, P. Nakkiran, and T. N. Sainath (2015). "Automatic gain control and multi-style training for robust small-footprint keyword spotting with deep neural networks". In: *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4704–4708. DOI: 10.1109/ICASSP.2015.7178863 (see pp. 8, 30, 75).

Prabhavalkar, Rohit, Kanishka Rao, Tara N. Sainath, Bo Li, Leif Johnson, and Navdeep Jaitly (2017). "A Comparison of Sequence-to-Sequence Models for Speech Recognition". In: *Proc. of Interspeech*. Stockholm, Sweden, pp. 939–943. DOI: 10.21437/Interspeech.2017-233 (see pp. 23, 76).

Qian, Yanmin, Tian Tan, Dong Yu, and Yu Zhang (2016). "Integrated adaptation with multi-factor joint-learning for far-field speech recognition". In: *Proc. of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. Shangai, China: IEEE, pp. 5770–5774. ISBN: 978-1-4799-9988-0. DOI: 10.1109/ICASSP.2016.7472783 (see p. 49).

Rabiner, L. R. (1989). "A tutorial on hidden Markov models and selected applications in speech recognition". In: *Proceedings of the IEEE* 77.2, pp. 257–286. ISSN: 0018-9219. DOI: 10.1109/5.18626 (see p. 13).

Rao, K., H. Sak, and R. Prabhavalkar (2017). "Exploring architectures, data and units for streaming end-to-end speech recognition with RNN-transducer". In: *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pp. 193–199. DOI: 10.1109/ASRU.2017.8268935 (see p. 25).

Ravi, Sachin and Hugo Larochelle (2017). "Optimization as a Model for Few-Shot Learning". In: *Proc. of the International Conference on Learning Representations (ICLR)*. Toulon, France (see p. 75).

Robinson, A. J. (1994). "An application of recurrent nets to phone probability estimation". In: *IEEE Transactions on Neural Networks* 5.2, pp. 298–305. ISSN: 1045-9227. DOI: 10.1109/72.279192 (see p. 17).

Rosenberg, A., K. Audhkhasi, A. Sethy, B. Ramabhadran, and M. Picheny (2017). "End-to-end speech recognition and keyword search on low-resource languages". In: *Proc. of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. New Orleans, LA, USA, pp. 5280–5284. DOI: 10.1109/ICASSP.2017.7953164 (see pp. 23, 28, 75).

Ruder, Sebastian (2016). "An overview of gradient descent optimization algorithms". In: *arXiv:1609.04747 [cs]*. arXiv: 1609.04747 (see p. 22).

Rumelhart, David E., Geoffrey E. Hinton, and Ronald J. Williams (1986). "Learning representations by back-propagating errors". In: *Nature* 323.6088, pp. 533–536. ISSN: 0028-0836. DOI: 10.1038/323533a0 (see p. 20).

Sainath, T. N., B. Kingsbury, A. Mohamed, and B. Ramabhadran (2013). "Learning filter banks within a deep neural network framework". In: *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*, pp. 297–302. DOI: 10.1109/ASRU.2013.6707746 (see p. 9).

Sainath, Tara N and Carolina Parada (2015). "Convolutional Neural Networks for Small-Footprint Keyword Spotting". en. In: *Proc. of Interspeech*. Dresden, Germany, p. 5 (see pp. 30, 80).

Sak, Haşim, Andrew Senior, and Françoise Beaufays (2014). "Long short-term memory recurrent neural network architectures for large scale acoustic modeling". In: *Fifteenth annual conference of the international speech communication association* (see p. 17).

Saon, G., H. Soltau, D. Nahamoo, and M. Picheny (2013). "Speaker adaptation of neural network acoustic models using i-vectors". In: *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*. Olomouc, Czech Republic, pp. 55–59. DOI: 10.1109/ASRU.2013.6707705 (see pp. 49, 50).

Saon, George, Gakuto Kurata, Tom Sercu, Kartik Audhkhasi, Samuel Thomas, Dimitrios Dimitriadis, Xiaodong Cui, Bhuvana Ramabhadran, Michael Picheny, Lynn-Li Lim, Bergul Roomi, and Phil Hall (2017). "English Conversational Telephone Speech Recognition by Humans and Machines". In: *arXiv:1703.02136 [cs]*. arXiv: 1703.02136 (see p. 49).

Savran, Arman, Raffaele Tavarone, Bertrand Higy, Leonardo Badino, and Chiara Bartolozzi (2018). "Energy and Computation Efficient Audio-visual Voice Activity Detection Driven by Event-cameras". In: *Proceedings of 13th IEEE International Conference on Automatic Face Gesture Recognition*. Xi'an, China (see pp. x, 102, 113).

Schwartz, R., Y. Chow, O. Kimball, S. Roucos, M. Krasner, and J. Makhoul (1985). "Context-dependent modeling for acoustic-phonetic recognition of continuous speech". In: *ICASSP '85. IEEE International Conference on Acoustics, Speech, and Signal Processing*. Vol. 10, pp. 1205–1208. DOI: 10.1109/ICASSP.1985.1168283 (see p. 14).

Seide, F., G. Li, X. Chen, and D. Yu (2011). "Feature engineering in Context-Dependent Deep Neural Networks for conversational speech transcription". In: *2011 IEEE Workshop on Automatic Speech Recognition Understanding*, pp. 24–29. DOI: 10.1109/ASRU.2011.6163899 (see p. 50).

Seltzer, M.L., Dong Yu, and Yongqiang Wang (2013). "An investigation of deep neural networks for noise robust speech recognition". In: *Proc. of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 7398–7402. DOI: 10.1109/ICASSP.2013.6639100 (see pp. 48, 49).

Senior, A., G. Heigold, M. Bacchiani, and H. Liao (2014). "GMM-free DNN acoustic model training". In: *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5602–5606. DOI: 10.1109/ICASSP.2014.6854675 (see p. 15).

Senior, A. and I. Lopez-Moreno (2014). "Improving DNN speaker independence with I-vector inputs". In: *Proc. of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. Florence, Italy, pp. 225–229. DOI: 10.1109/ICASSP.2014.6853591 (see p. 49).

Sennrich, Rico, Barry Haddow, and Alexandra Birch (2016). "Neural Machine Translation of Rare Words with Subword Units". en. In: *Proc. of the 54th Annual Meeting of the Association for Computational Linguistics* (see p. 25).

Shan, Changhao, Junbo Zhang, Yujun Wang, and Lei Xie (2018). "Attention-based End-to-End Models for Small-Footprint Keyword Spotting". In: *arXiv:1803.10916 [cs, eess]*. arXiv: 1803.10916 (see pp. 30, 74, 75, 78, 93).

Shinohara, Yusuke (2016). "Adversarial Multi-Task Learning of Deep Neural Networks for Robust Speech Recognition". In: *Proc. of Interspeech*. San Francisco, CA, USA, pp. 2369–2372 (see p. 49).

Snell, Jake, Kevin Swersky, and Richard Zemel (2017). "Prototypical networks for few-shot learning". In: *Advances in Neural Information Processing Systems*. Long Beach, CA, USA, pp. 4077–4087 (see p. 75).

Soltau, H., H. Liao, and H. Sak (2017). "Reducing the computational complexity for whole word models". In: *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pp. 63–68. DOI: 10.1109/ASRU.2017.8268917 (see pp. 25, 29).

Soltau, Hagen, Hank Liao, and Hasim Sak (2016). "Neural Speech Recognizer: Acoustic-to-Word LSTM Model for Large Vocabulary Speech Recognition". In: *arXiv:1610.09975 [cs]*. arXiv: 1610.09975 (see pp. 25, 29).

Swietojanski, Pawel and Steve Renals (2014). "Learning hidden unit contributions for unsupervised speaker adaptation of neural network acoustic models". In: *Spoken Language Technology Workshop (SLT), 2014 IEEE*. IEEE, pp. 171–176 (see p. 51).

Swietojanski, Pawel and Steve Renals (2016). "SAT-LHUC: Speaker adaptive training for learning hidden unit contributions". In: *Proc. of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. IEEE, pp. 5010–5014 (see p. 51).

Szöke, Igor, Petr Schwarz, Pavel Matějka, and Martin Karafiát (2005). "Comparison of keyword spotting approaches for informal continuous speech". In: *In Proceedings Eurospeech* (see pp. 27, 30).

Tang, R. and J. Lin (2018). "Deep Residual Learning for Small-Footprint Keyword Spotting". In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5484–5488. DOI: 10.1109/ICASSP.2018.8462688 (see p. 30).

Tjandra, Andros, Sakriani Sakti, and Satoshi Nakamura (2017). "Local Monotonic Attention Mechanism for End-to-End Speech and Language Processing". en. In: *Proc. of the 8th International Joint Conference on Natural Language Processing*. arXiv: 1705.08091. Taipei, Taiwan (see p. 24).

Tóth, L. (2014). "Combining time- and frequency-domain convolution in convolutional neural network-based phone recognition". In: *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 190–194. DOI: 10.1109/ICASSP.2014.6853584 (see p. 17).

Vapnik, V. and A. Vashist (2009). "A new learning paradigm: learning using privileged information." eng. In: *Neural networks : the official journal of the International Neural Network Society* 22.5-6, pp. 544–557. ISSN: 0893-6080. DOI: 10.1016/j.neunet.2009.06.042 (see p. 53).

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin (2017). "Attention is All you Need". In: *Advances in Neural Information Processing Systems (NIPS)*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Curran Associates, Inc., pp. 5998–6008 (see p. 23).

Veselỳ, Karel, Arnab Ghoshal, Lukás Burget, and Daniel Povey (2013). "Sequence-discriminative training of deep neural networks." In: *Interspeech*, pp. 2345–2349 (see p. 19).

Vincent, Emmanuel, Shinji Watanabe, Aditya Arie Nugraha, Jon Barker, and Ricard Marxer (2017). "An analysis of environment, microphone and data simulation mismatches in robust speech recognition". In: *Computer Speech & Language*. ISSN: 08852308. DOI: 10.1016/j.csl.2016.11.005 (see pp. 36, 38).

Viterbi, A. (1967). "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm". In: *IEEE Transactions on Information Theory* 13.2, pp. 260–269. ISSN: 0018-9448. DOI: 10.1109/TIT.1967.1054010 (see p. 16).

Warden, Pete (2018). "Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition". In: *arXiv:1804.03209 [cs]*. arXiv: 1804.03209 (see pp. 36, 39, 74, 80).

Watanabe, S., T. Hori, J. Le Roux, and J. R. Hershey (2017). "Student-teacher network learning with enhanced features". In: *Proc. of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 5275–5279. DOI: 10.1109/ICASSP.2017.7953163 (see p. 53).

Watanabe, Shinji, Takaaki Hori, S. Kim, J. R. Hershey, and T. Hayashi (2017). "Hybrid CTC/Attention Architecture for End-to-End Speech Recognition". In: *IEEE Journal of Selected Topics in Signal Processing* 11.8, pp. 1240–1253. ISSN: 1932-4553. DOI: 10.1109/JSTSP.2017.2763455 (see pp. 25, 36).

Werbos, Paul J. (1988). "Generalization of backpropagation with application to a recurrent gas market model". In: *Neural Networks* 1.4, pp. 339–356. ISSN: 0893-6080. DOI: 10.1016/0893-6080(88)90007-X (see p. 17).

Wu, Yonghui, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean (2016). "Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation". en. In: (see p. 25).

Xiong, W., J. Droppo, X. Huang, F. Seide, M. Seltzer, A. Stolcke, D. Yu, and G. Zweig (2016). "Achieving Human Parity in Conversational Speech Recognition". In: *arXiv:1610.05256 [cs]*. arXiv: 1610.05256 (see pp. 1, 18).

Yang, Flood Sung Yongxin, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales (2018). "Learning to compare: Relation network for few-shot learning". In: *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Salt Lake City, UT, USA (see p. 75).

Young, S. J., J. J. Odell, and P. C. Woodland (1994). "Tree-based State Tying for High Accuracy Acoustic Modelling". In: *Proceedings of the Workshop on Human Language Technology*. HLT '94. event-place: Plainsboro, NJ. Stroudsburg, PA, USA: Association for Computational Linguistics, pp. 307–312. ISBN: 978-1-55860-357-8. DOI: 10.3115/1075812.1075885 (see p. 14).

Young, S. J. and P. C. Woodland (1994). "State clustering in hidden Markov model-based continuous speech recognition". In: *Computer Speech & Language* 8.4, pp. 369–383. ISSN: 0885-2308. DOI: 10.1006/csla.1994.1019 (see p. 14).

Young, Steve, Gunnar Evermann, Mark Gales, Thomas Hain, Dan Kershaw, Xunying Liu, Gareth Moore, Julian Odell, Dave Ollason, Dan Povey, Anton Ragni, Valtcho Valtchev, Phil Woodland, and Chao Zhang (2015). *The HTK book (for HTK version 3.5)*. Cambridge University Engineering Department (see pp. 34, 35).

Yu, Dong and Li Deng (2015). *Automatic Speech Recognition: A Deep Learning Approach*. en. Signals and Communication Technology. London: Springer-Verlag. ISBN: 978-1-4471-5778-6 (see p. 49).

Yu, Dong, Michael L. Seltzer, Jinyu Li, Jui-Ting Huang, and Frank Seide (2013). "Feature Learning in Deep Neural Networks - Studies on Speech Recognition Tasks". In: *Proc.*

of the International Conference on Learning Representations (ICLR). arXiv: 1301.3605 (see p. 48).

Yu, Dong, Kaisheng Yao, Hang Su, Gang Li, and Frank Seide (2013). "KL-divergence regularized deep neural network adaptation for improved large vocabulary speech recognition". In: *Proc. of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. IEEE, pp. 7893–7897. ISBN: 978-1-4799-0356-6. DOI: 10.1109/ICASSP.2013.6639201 (see p. 51).

Yu, J., K. Markov, and T. Matsui (2016). "Articulatory and spectrum features integration using generalized distillation framework". In: *2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP)*, pp. 1–6. DOI: 10.1109/MLSP.2016.7738813 (see p. 53).

Zeiler, Matthew D. (2012). "ADADELTA: An Adaptive Learning Rate Method". In: *arXiv:1212.5701 [cs]*. arXiv: 1212.5701 (see p. 21).

Zeyer, Albert, Kazuki Irie, Ralf Schlüter, and Hermann Ney (2018). "Improved training of end-to-end attention models for speech recognition". In: *arXiv:1805.03294 [cs, stat]*. arXiv: 1805.03294 (see p. 25).

Zeyer, Albert, Ralf Schlüter, and Hermann Ney (2016). "Towards Online-Recognition with Deep Bidirectional LSTM Acoustic Models." In: *Proc. of Interspeech*, pp. 3424–3428 (see p. 26).

Zhang, C. and P. C. Woodland (2014). "Standalone training of context-dependent deep neural network acoustic models". In: *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5597–5601. DOI: 10.1109/ICASSP.2014.6854674 (see p. 15).

Zhang, Chao and Philip C. Woodland (2015). "A general artificial neural network extension for HTK." In: *Proc. of Interspeech*. Dresden, Germany (see p. 35).

Zhuang, Yimeng, Xuankai Chang, Yanmin Qian, and Kai Yu (2016). "Unrestricted Vocabulary Keyword Spotting Using LSTM-CTC." In: *INTERSPEECH*, pp. 938–942 (see p. 28).

Appendix A

Grammars

All grammars are represented using HTK grammar definition language. It consists of a set of variables (starting with $) which are defined by a regular expression. Vertical bars (|) indicate alternatives. Special symbols SENT-START and SENT-END represent sentence start and end.

A.1 VoCub original grammar

Original grammar for the VoCub dataset, corresponding to the commands of the IOL demo. Table A.1 gives the list of 15 commands that were recorded twice per speaker.

```
$vObject = octopus;
$cObject = toy | lego | ladybug | turtle | car | bottle | box;
$object = $vObject | $cObject;
$aObject = an $vObject | a $cObject;
$side = left | right;
$action = Take | Grasp | Touch | Push | Explore;
$sent = Skip it |
        Yes you are |
        No you are not |
        No here it is |
        Yes I do |
        No I do not |
        There you go |
        Finished |
        Return to home position |
```

```
        Calibrate on table |
        See you soon |
        I will teach you a new object |
        Execute a plan |
        What is this |
        Wrong this is $aObject |
        This is $aObject |
        Where is the $object |
        $action the $object |
        Let me show you how to reach the $object with your $side arm |
        Forget the $object |
        Forget all objects;
( SENT-START ( $sent ) SENT-END )
```

TABLE A.1 List of the 15 commands that were recorded twice per speaker for the VoCub dataset.

Skip it.
Yes you are.
No you are not.
No here it is.
Yes I do.
No I do not.
There you go.
Finished.
Return to home position.
Calibrate on table.
See you soon.
I will teach you a new object.
Execute a plan.
What is this?
Forget all objects.

A.2 VoCub extended grammar

Extended grammar for the VoCub dataset, corresponding to the original commands of the IOL demo and 20 additional sentences recorded for the test set only.

```
$vObject = octopus;
$cObject = toy | lego | ladybug | turtle | car | bottle | box;
$object = $vObject | $cObject;
$aObject = an $vObject | a $cObject;
$side = left | right;
$action = Take | Grasp | Touch | Push | Explore;
$objDemo = keys | pen;
$sent = Skip it |
        Yes you are |
        No you are not |
        No here it is |
        Yes I do |
        No I do not |
        There you go |
        Finished |
        Return to home position |
        Calibrate on table |
        See you soon |
        I will teach you a new object |
        Execute a plan |
        What is this |
        Wrong this is $aObject |
        This is $aObject |
        Where is the $object |
        $action the $object |
        Let me show you how to reach the $object with your $side arm |
        Forget the $object |
        Forget all objects |
        Hold the glass |
        Go back to initial state |
        False that is a cup |
        Analyze the dolphin |
        Let me teach you how to pick up the can with your left hand |
        Talk to you later |
        Let me teach you how to pick up the tool with your right hand |
        Drop that ball |
        Give me some coins |
```

```
        Turn the phone |
        Let me demonstrate how to grab the $objDemo with your other hand |
        Hold on |
        Stop it |
        False this is not a spoon |
        Accomplish the task |
        Done |
        That is a pair of scissors |
        Pull the hammer |
        Shake this bag;
( SENT-START ( $sent ) SENT-END )
```

A.3 Reduced grammar for the online detection demo

Reduced grammar used for the online detection demo with R1.

```
$vObject = octopus;
$cObject = toy | lego | ladybug | turtle | car | bottle | box;
$object = $vObject | $cObject;
$aObject = an $vObject | a $cObject;
$sent = Yes you are |
        No you are not |
        Return to home position |
        See you soon |
        What is this |
        This is $aObject |
        Where is the $object |
        Forget the $object |
        Forget all objects;
( SENT-START ( $sent ) SENT-END )
```